Expressive and Explainable Rule-Based Classification

MANAL KHALAF ALMUTAIRI

A thesis submitted in partial fulfilment of the requirements of University of Reading for the degree of Doctor of Philosophy in Computer Science

Department of Computer Science

NOVEMBER 2021

Abstract

Classification rule learning produces expressive rules so that a human user can easily interpret the rationale behind the predictions of the generated model. Constructing a very accurate classification model may lead to overfitting, a common problem in data mining that causes a leaner to perform badly on test instances. Ensemble learning is a common way to address the problem of overfitting while improving the learner's accuracy. The idea of ensemble classification is to construct various predictive base learners, and then, combine their predictions. This often goes at the expense of explainability of the predictive model learned, as the analyst is presented with many different classification models. Therefore, predictive learning models are required to be not only reliable and accurate, but also comprehensible to avoid the risk of irreversible misclassification, especially in critical applications such as medical diagnoses, financial analysis, terrorism detection, etc. The level of expressiveness of the individual base learners is one of the most important factors for improving the whole ensemble's explainablility.

Taking this into account, this research focuses on developing a predictive ensemble learner that maintains the expressive power of rule learning models while benefiting from the high predictive performance of ensemble learning. Measuring the expressiveness of a rule-based learner often depends on the complexity of its rule set. A rule set is considered more expressive when it produces fewer number of rules with less complex terms per rule. Also, rule learning approaches can abstain from classification when the algorithm is uncertain about a prediction, which contributes to increased explainablility in the model by ensuring the trustworthiness of the induced rule set. Abstaining is needed to prevent costly false classification. Nevertheless, classifying instances correctly is more desired than abstaining from it in most applications. Therefore, this thesis aims to answer the following raised research question: *'is it possible to develop a predictive ensemble model, which exhibits a similar expressiveness as the predictive base learner while improving its accuracy and lowering its abstaining rate?'*.

To achieve that, this thesis makes a number of contributions towards rule induction algorithms in both single-based and ensemble-based systems. Three novel single predictive rule-based algorithms are developed, termed G-Prism-FB, G-Prism-DB, and G-Rules-IQR, where 'G' stands for Gaussian distribution. These algorithms are highly expressive on their own, that can be used to serve as the base learners of the ensemble. The results of empirical evaluation show that these

algorithms produce expressive and more computationally efficient numeric rule-terms compared with frequent discrete intervals. G-Rules-IQR learner, in particular, has shown to be superior in terms of expressiveness and accuracy compared with other rule-based learners. Therefore, it is utilised in this research as a base learning algorithm to induced multiple base classifiers for the ensemble systems.

Furthermore, a novel framework for explainable rule-based ensemble algorithms called ReG-Rules (Ranked ensemble G-Rules) is presented. ReG-Rules incorporates three novel methods. First, a new ranking-based approach to rank the base classifiers, and then a selection method to find the best performing models. Second, a new rules merging algorithm to reduce the number of rules induced by each selected model without loss of rule coverage. Third, to reduce the overall number of rules presented to human analyst during the prediction stage, a decision committee of rules is built per classification attempt using a novel weighted voting combination method.

Additionally, an extension of ReG-Rules learner termed CRC (Consolidated Rules Construction) is developed. CRC enhances the explainablility of ReG-Rules by generating rules that can be consolidated into a single global rule set and used directly in predictions without the need for a classification committee. The experimental studies show that comparred with the standalone classifier (G-Rues-IQR), ReG-Rules and CRC, are more accurate on all cases. Also, the ReG-Rules and CRC abstaining rates were almost zero on all cases, while the abstaining rate of G-Rules-IQR learner was above 10% in several cases.

Dedication

I dedicate this thesis to...

My great parents,
My beloved husband,
My wonderful children

ACKNOWLEDGEMENTS

First and foremost, praises and thanks to the God, the almighty, for his showers of blessings throughout my research work.

I would like to express my deepest and sincerest gratitude to my research supervisor, Dr. Frederic Stahl for his constant support, patience and constructive comments that have greatly improved this work. I was very fortunate by his supervision and encouragement throughout the journey of my PhD. Without his expertise and countless number of discussions the completion of this thesis would not have been possible. I would also like to thank Prof. Max Bramer for his valuable advices. I appreciate the time he spent for reviewing the published work described in this thesis. Thanks to my both assessors Prof. Xia Hong and Dr. Hong Wie for their feedback during the annual reviews of the PhD project.

A special thanks and sincere gratitude goes to my beloved family. Words can not express how grateful I am for all the sacrifices that you have made on my behalf. Thanks to my loving parent whose love and affection is the source motivation and encouragement of my studies. Thank you to my wonderful husband for being patient, supportive and helpful at every stage of my study. Thanks to my children for their understanding and patience throughout PhD years. Thank you to my brothers and sisters for simply being there when needed.

I would also like to extend my thanks to all my fellow researchers, especially those from data science lab for their useful discussions and helpful advice.

Last but not least, I acknowledge deep gratitude to my country, the kingdom of Saudi Arabia, for giving me the PhD scholarship and financial support for this research.

Manal Khalaf Almutairi
November 2021

Original Authorship

Declaration: I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Name: Manal Khalaf Almutairi

Sign: ———————————-

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Predictive Ensemble Learning

In machine learning and data mining, *classification* is a popular supervised learning approach. The process aims to derive a model that describes the hidden patterns labelled by target classes (concepts) in the training data and then uses this learned model (classifier) to accurately predict the label of a new instance in the testing data [1, 2]. Each instance is represented by a number of input *attributes*, also known as *features*. Training and testing stages are also known as *learning* and *prediction* respectively.

Predictive learning models are required to be accurate and also comprehensible to reduce the risk of irreversible wrong classification. This is especially the case in many critical applications, such as medical diagnoses, financial analysis, credit risk evaluation, terrorism detection, etc. where the predictive model should explain the reason for classification to the decision makers, and not only produce predictions. On the other hand, to build an accurate and efficient model, it is important to avoid the problems of overfitting and underfitting.

'Overfitting', which is a common problem in learning algorithms happens when a model is over-optimised for the training data. This is often caused by noise or outliers in the dataset and it is likely to result in a learner with high variance that achieves a very good fit to the training data, but performs badly on the test data. Variance is the variability of model prediction for a given class label. Overfitting is also, a major cause of constructing a complex predictive learner to the human analyst, which may considerably reduce the comprehensibility of the learner [1].

In terms of 'underfitting' problem, it happens when a model is over-simplified and pays very little attention to the training data. This leads to a learner with high bias that often performs badly on training and test data. Bias is the difference between the average prediction of a model and the correct value which the model

are trying to predict [1].

In most predictive learning algorithms, the standard strategy to avoid overfitting is simplifying/generalising the model, i.e. sacrificing accuracy on the training set for accuracy of classifying unseen data. This trade-off in complexity is a trade-off between bias and variance with the aim of finding a good balance without overfitting and undefitting. However, there is no ideal learning algorithm that can successfully use this trade-off strategy on all types of datasets [3]. Therefore, the alternative common approach which has initially been developed in order to reduce overfitting and underfitting, while improving the predictive accuracy, is ensemble learning [4]. This can be explained by the main concept and philosophy of ensemble classification which is based on learning not just one classifier, but various base classifiers induced from diverse samples of the training data [5]. Then, combining the predictions of several classifiers can efficiently remove the high variance or high bias that may exist in predictions and perform better than individual models that may overfit/underfit the training data [6]. The prediction is usually derived through a voting strategy, i.e. majority, weighted majority voting, etc. Chapter 2 explains why ensembles work better, on average, than single base learners used by the ensemble.

However, the use of ensemble approaches defies the purpose of explainability, as the human is presented with a large range of entire classification models, such as multiple decision trees. This would challenge the ability of decision makers to understand how a predictive ensemble system makes its decisions as the human analyst would have to examine many decision models to gain insights about the causality of the prediction. Therefore, this thesis is concerned with developing and implementing a predictive ensemble learning model that is interpretable by humans while retaining key advantages of ensembles learners.

1.2 Research Motivations

Despite that the predictive accuracy is considered to be the key evaluation criterion in most classification methods, the importance of comprehensible learning models is considerably increasing in many application domains, such as medicine and banking [7]. In regard to the accuracy, the main concept of ensembles is based on the idea that combining several individual learners would often generate a learner with a higher predictive accuracy than its components [1, 3, 5, 8–10]. However, concerning expressiveness, most ensemble learners are hardly readable by a human [2, 3, 7].

Therefore, this thesis is mainly motivated by the desire to develop a predictive ensemble learner that can be both accurate and expressive at the same time. This

will include the transformation of the proposed ensemble classification model into a consolidated ensemble learner that can be more expressive while preserving the predictive accuracy of the ensemble it was derived from.

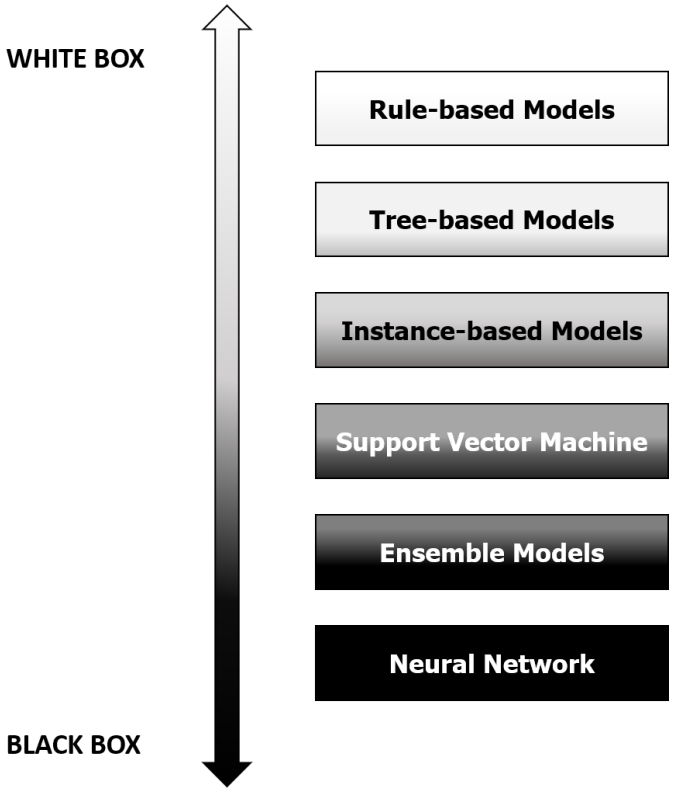


Figure 1.1: Hierarchy of output expressiveness (source: adapted from [11])

Various machine learning algorithms exist to induce classification models that can be used as base classifiers for a predictive ensemble learner. However, it was of interest in this study to only investigate approaches that share the common goal of producing models that are readable by humans. This means excluding all the black box learning approaches such as Support Vector Machine (SVM), Kernel-based learning, Artificial Neural Networks (ANN) and Deep Learning approach. It also means not using the Tree-based models, which are not sufficiently expressive as they tend to be complex once the tree grows to a certain size, and hence they become increasingly difficult for humans to understand and maintain. Consequently, *the project focuses solely on modular Rule Induction approaches as they offer a greater explainability about how they arrive at a particular prediction compared with Decision Trees*. They are much closer to the white box models than other techniques, as illustrated in Figure 1.1. Rule-based predictive methods use a set of IF-THEN rules for classification [1, 3]. Each rule is an expression of the form:

IF condition(s) THEN conclusion

The left-hand-side (LHS) of a rule is known as *precondition* while the right-hand-side (RHS) is the *rule consequent*. A verity of Rule Induction algorithms were

investigated in this research. A review of them is provided in Chapter 2.

The terms explainable and expressive are similar, but there is a subtle semantic difference how they are used in this thesis. The term explainability refers to classification models that explain the outcome of a predicted label to the analyst. The less information is needed to explain the model, the higher the degree of explainability. Similarly, the term expressive is used in this thesis in the context of single rules. A rule is more expressive the more compact the information leading to a prediction is encoded in the rule. The research presented in this thesis focuses on the explainability aspect of ensemble classifiers by minimising the amount of rules needed to derive a prediction. However, on a rule level also, the most expressive types of rules are utilised.

1.3 Research Question, Hypothesis and Objectives

The aim of this thesis is to answer the following research question:

Is it possible to develop a predictive ensemble model, which exhibits a similar expressiveness as the predictive base learner while improving its accuracy and lowering its abstaining rate?

To answer the research question, this thesis focuses on a number of investigations, experimental studies and empirical evaluations that involve discussing/testing the following proposition/hypothesis:

> *We can develop a predictive ensemble learning system using an expressive rule-based algorithm or similar white box approach as a base learners' inducer, and this system will be more accurate than its components while producing compact and human-readable rules.*

The following research objectives would facilitate the testing of the aforementioned hypothesis, which may lead ultimately to the achievement of the research aim:

Objective 1: To critically assess rule-based and ensembles predictive techniques and their limitations.

Objective 2: To measure and compare the expressiveness of rule based models and develop an appropriate rule-based predictive algorithm suitable as base learner for an ensemble.

Objective 3: To improve the quality of rule sets by developing rule merging techniques for predictive rules and minimising loss of accuracy.

Objective 4: To develop an expressive ensemble learner footed upon the base classifier developed in objective 2 and the rule merging techniques in objective 3.

1.4 Research Methodology

This theoretical section focuses on reflecting the nature of this research and the required methods to tackle the research area properly. It uses the 'research onion' (Figure 1.2) as a way of describing the research methodology. Briefly, according to [12], the research onion consists of the following layers:

1. *The main philosophy and paradigm* - forms the philosophical stance of the research. Amongst the two common paradigms that have been adopted in this thesis (positivism and interpretivism), which will be briefly discussed in the next section.

2. *The logical approaches to theory development* - can be implied by the research philosophy on previous layer and usually includes: deduction and induction. The former approach is utilised in the current project. The differences between them will be explained in the next section.

3. *Methodological choice* - this layer specifies the use of quantitative, qualitative methods or combinations of both, which is the choice adopted in this work. Further details are in Section 1.4.2.

4. *Strategies* - this layer includes: experiment, survey, quasi-experiment, case study, ethnography, grounded theory, etc. This thesis uses experiment and case study strategies to answer the research question. Section 1.4.2 and Section 1.4.4 explain how these two strategies are utilised in this project.

5. *Techniques and procedures* - include data collection and data analysis. Section 1.4.3 determines the data sources used in this research.

1.4.1 Research Philosophy and Logical Approaches

The main goal of any research is generalising results that obtained from some adopted strategies to the extent that they could be applied beyond the area under

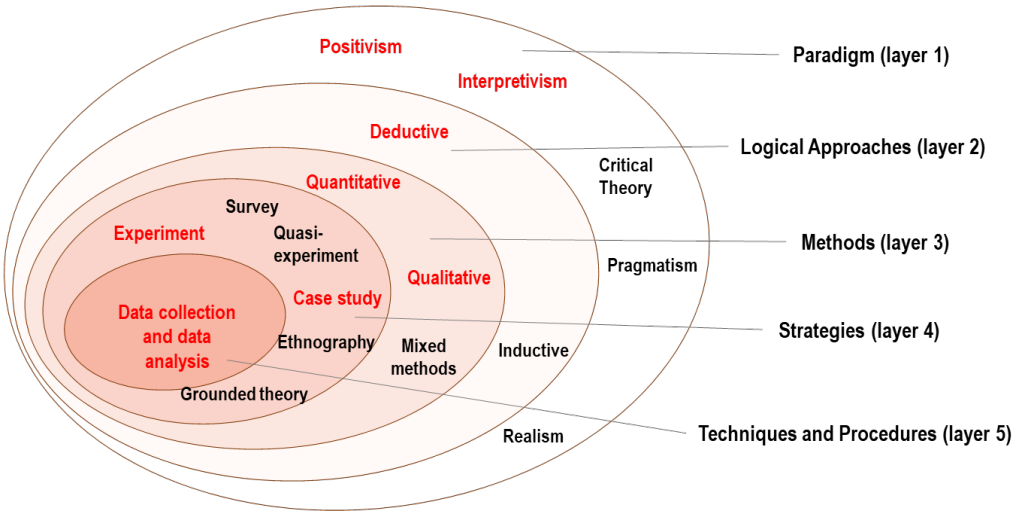


Figure 1.2: The research onion - red colour refers to the ways adopted in this thesis (source: Adapted from Saunders et al. 2015)

investigation [13, 14]. However, the system of ideas, beliefs and assumptions that are utilised by a community of researchers to obtain knowledge from the world differs significantly between communities depending on the origins and the characteristics of each field. As can be seen in the research onion, layer 1 includes a number of philosophies (paradigms). However the current section focuses on positivism and interpretativism because they are to some extent related to the strategies adopted in the research presented in this thesis. For detailed descriptions of all research paradigms, the reader is referred to [14].

The *positivism paradigm* is a methodological philosophy often involves the use of present theory to develop hypotheses to be tested during the research process. In this respect, any conclusion needs to be measured using trustworthy measurements and statistics [12]. Often, positivism prefers to quantitative methods to extract new knowledge from positive interpretation of results using experiments.

The *interpretivism paradigm* is a methodological philosophy focus primarily on understanding of knowledge related to human subjective experience and social sciences [14]. Consequently, according to this paradigm the research outcomes may have more than one interpretation in contrast to the positivism paradigm where the findings of the research focus on only one possible explanation [12]. To understand the knowledge, interpretivism rely on qualitative methods such as case studies. Please note that both paradigms are utilised in this research.

Regarding the logical approaches to theory development (layer 2), two contrasting approaches are outlined here: *deductive* and *inductive*. The former moves from general to specific, i.e. the research develops the hypothesis or hypotheses upon a pre-existing theory, which is often developed from academic literature. Then the research designs a strategy to test this theory. The deductive approach

thus might be considered particularly suited to the quantitative methods. However, qualitative research methods can also be utilised under the deductive reasoning approaches. In contrast, inductive reasoning moves from specific to general, i.e. the observations are the starting point for the researcher. Then as the data is analysed, it may be found that it can be fit into an existing theory or even new theories can be generated. Please note that this research follows the inductive reasoning approach as it is based on analysing data and observations in order to generate new theories (approaches).

1.4.2 Research Methods and Strategies

Considering layers 3 and 4 in Figure 1.2, the underlying hypothesis in machine learning field is relatively close to the positivism philosophy (quantitative methods) and therefore, experimental studies are heavily relied on to verify or even falsify the hypothesis on empirical data. Nevertheless, the interpretivism (qualitative methods) is considered to be a common methodological philosophy in machine learning research. There are also well-established methods, which are suitable and specific to Computer Science discipline [15].

Roughly speaking, machine learning research is typically based on: (i) *theoretical approaches* like logic, mathematics and algorithms. (ii) *practical approaches* like implementations, systems, or combination of both. In computer science discipline, this involves implemented the machine learning algorithms as software (written code) solutions in which their performance can be empirically evaluated and compared to other solutions [16].

As mentioned in the previous section, this project is mainly under the positivist paradigm as most of the methods adopted in this thesis to answer the research question are quantitative (experiments strategies in particular). However, part of the evaluation studies presented in chapter 5 are conducted under the interpretivism paradigm in which a qualitative method was used in the form of case study strategy.

To sum up, multi-methods strategy is employed in this research. The strategy involves: (1) proposing novel algorithms from an existing mathematical concept (theoretical method), (2) implementing the proposed algorithms using R statistical language (practical method), and (3) evaluating the algorithms using (i) quantitative methods for the most proposed algorithms (experimental studies) and (ii) qualitative methods (case studies) for algorithms proposed in Chapter 5.

1.4.3 Data Sources

Referring to layer 5 in Figure 1.2, the datasets used in this thesis are taken from well-known Repositories like UCI ML [17] and KDD [18], which are widely used datasets for benchmarking machine learning algorithms from several problem domains with many different characteristics. The detailed descriptions of the datasets that have been used in this thesis are provided in each experimental study independently. Noticeably, all the datasets are chosen randomly and the only condition being that they contain continuous attributes and involve classification tasks.

1.4.4 Evaluation Procedures

The final and the most important stage in any machine learning project is the evaluation process of the model. This is defined by what the problem scope and what the research aim are. A suitable metric, or a set of metrics must be carefully selected to provide the best assessment of how well a system meets the criteria of the evaluation process. As aforementioned, the central focus in this project is to produce correct predictions using predictive rules with the ability to be reviewed and manually investigated by the decision maker. Taking this into account, the concentration will be on maximising the expressive power of the system while improving its predictive accuracy. The former will be assessed by how well the model can generate rules that are fully comprehensible to a human, and the latter will be measured by the quality and the accuracy of these rules. Therefore, all the algorithms this project developed were quantitatively evaluated using several experimental studies.

In addition, some of the methods proposed in this thesis were also qualitatively evaluated based on a number of case studies. To estimate the accuracy of the classification models, two methods are adopted in this thesis. The first procedure is the 'holdout' where the given dataset is randomly partitioned into two independent sets, a training set, which is used to construct the model and a test set which is used to estimate the accuracy of the model [1]. The second procedure is 'Bagging', which is a popular method introduced by [9], and used to improve the stability of the model by partitioning the data several times with replacement. Each sample is likely to have approximately 63.2% of the instances which can be used to train a base model, while the remaining (about 36.8%) which called out-of-bag (OOB) instances can be used to validate the base model performance.

1.5 Contributions to Knowledge

This thesis makes a number of contributions towards rule induction algorithms in both single-based and ensemble-based systems. The contributions can be summarised as follows:

1. A novel classification rule induction approach, called G-Prism, which makes use of Gauss Probability Density Distribution (GPDD) and z-score distribution to generate a new rule-term structure that improves classification performance of the PRISM family of algorithms. The new approach produces more expressive and computationally efficient numeric rule-terms compared with converting continuous attributes into categorical ones in the form of frequent discrete intervals. The first version of G-Prism was termed **(G-Prism-FB)** and it has been introduced in [19], a paper published in the 36^{th} SGAI International Conference on Artificial Intelligence.

2. A novel classification rule induction approach, called **G-Prism-DB**, which is a second version of G-Prism algorithm that enables it to expand the coverage of each numeric rule-term, and thus produces fewer rules which are less prone to overfitting. The algorithm is based on a new dynamic rule-term boundaries approach to improve the expressiveness of the rules induced. The dynamic sized boundary can be defined by the user. The work has been introduced in [20], a paper published in the 37^{th} SGAI International Conference on Artificial Intelligence.

3. A novel predictive rule induction algorithm, called **G-Rules-IQR**, which incorporates two new methods in its construction:

   - A new more efficient method in learning heuristics to induce numerical rule-terms directly from continuous attributes based on a combination of: GPDD function, quartiles, and Interquartile Range (IQR). The new method enables producing more compact, accurate and expressive rules using IQR boundaries instead of user defined boundaries.
   - A new way to address the challenges in assuming normally distributed attributes in the previous GPDD rule learning algorithms by reducing the skewness rate of numerical attribute values from the normal distribution. G-Rules-IQR algorithm incorporates a prior testing for normality for each attribute in the dataset before applying the approximate normal transformation on the attribute's values.

   The work has been introduced in [21], a paper published in the 17^{th} IEEE International Conference on Machine Learning and Applications.

4. A novel framework for ensemble rule-based system, called **Ranked Ensemble G-Rules (ReG-Rules)** learner, which utilises G-Rules-IQR algorithm as a learning algorithm for its base classifiers. This ensemble model provides an approach to harvest the predictive power of an ensemble learner, while maintaining several explainable aspects of rule-based predictive models. ReG-Rules incorporates three novel methods in its construction:

   - A new **Ranking-based method** to rank the base classifiers according to a number of criteria, not only to their accuracies.
   - A new **local Rule Merging (RM) technique** that can reduce the number of rules induced within each individual base classifier without sacrificing the overall predictive accuracy of the model. The approach can be considered as a useful aid in improving the quality of the induced rules and thus developing more expressive rule learners.
   - A new combination technique called **'ReG-Rules Committees'**, which makes use of a weighted voting strategy to decide the final ensemble predictions. The method addresses the potential problem of reliability when some base models are more reliable than others.

   This work has been introduced in [22], a paper published in IEEE Access Journal.

5. A significant extension of the ensemble ReG-Rules learner that can be more expressive while benefiting from the high predictive performance of ensemble learning compared with its stand-alone base classifiers. This system, which is called **'Consolidated Rules Construction (*CRC*)'** incorporates the following novel method in its construction:

   - A new rule consolidation approach, termed **'CRC Consolidator'**, which can compress multiple classifiers' rule sets into one global rule set that can be used directly in predictions without the need to build a committee of rules for each new classification attempt.

6. All the aforementioned algorithms have been implemented and empirically evaluated. All the source codes, which were written in the statistical programming language R, are available in online repository at - `https://github.com/ManalAlmutairi/PhD_Project_Codes/tree/v1.0.0` and are archived at - `https://doi.org/10.5281/zenodo.5557590` [23].

The publications that have been produced during this project are listed below:

- Almutairi, Manal, Frederic Stahl, Mathew Jennings, Thien Le, and Max Bramer. "Towards Expressive Modular Rule Induction for Numerical Attributes." In International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp. 229-235. Springer, Cham, 2016.
- Almutairi, Manal, Frederic Stahl, and Max Bramer. "Improving Modular Classification Rule Induction with G-Prism using Dynamic Rule Term Boundaries." In International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp. 115-128. Springer, Cham, 2017.
- Almutairi, Manal, Frederic Stahl, and Max Bramer. "A Rule-Based Classifier with Accurate and Fast Rule Term Induction for Continuous Attributes." In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 413-420. IEEE, 2018.
- Almutairi, Manal, Frederic Stahl, and Max Bramer. "ReG-Rules: An Explainable Rule-Based Ensemble Learner for Classification." IEEE Access 9 (2021): 52015-52035.

1.6 Structure of the Thesis

This thesis is organised as follows:

Chapter 2. Introduces essential background of predictive data mining algorithms and their limitations in the classification tasks for both paradigms: stand-alone and ensembles. Classification rule induction has been thoroughly investigated and the algorithms related to its two types: 'divide and conquer' and 'separate and conquer' have been described in depth and compared to each other with examples. Several ensemble methods characteristics have been thoroughly explored in this chapter. Also, the main quality criteria in the predictive ensemble learning evaluation is discussed in this chapter.

Chapter 3. Discusses and analyses a number of limitations existing in separate and conquer approaches, particularly in PRISM family of algorithms. The chapter focuses on challenges in dealing with numeric attributes and thus it thoroughly reviews a number of common (local and global) discretisation methods. The computational issues with discretising continuous attributes using local cut-point calculations in PRISM family of algorithms are investigated. The chapter introduces

an alternative and more efficient technique to handle numeric values based on Gaussian Probability Density Distribution (GPDD). The new technique has been utilised to develop two new members in PRISM family of algorithms, namely, G-Prism-FB and G-Prism-DB. The letter 'G' stands for Gaussian probability density distribution, 'FB' refers to fixed size rule-term boundaries and 'DB' stands for dynamic size rule-term boundaries. These algorithms are empirically evaluated in this chapter and compared with the original PRISM algorithm and against each other.

Chapter 4. Highlights some limitations that exist in G-Prism algorithms, then the solutions resulting in a new rule induction classifier called 'G-Rules-IQR' are presented. The chapter also, illustrates the methods integrated in G-Rules-IQR theoretically and empirically. The methods are a combination of GPDD technique, Interquartile Range (IQR), and a transformation approach towards normally distributed data. For comparative purposes, this chapter implements different versions of the original PRISM with various well-known discretisation methods (binary splitting, ChiMerge and Caim). Also, the implementations of G-Rules-IQR and G-Prism algorithms allowed to switch off the transformation to approximate normal distribution. This will be followed by empirical evaluation in which multiple comparisons between the aforementioned algorithms are conducted.

Chapter 5. Discusses and analyses a number of limitations related to stand-alone learning systems, which emphasise the strength of utilising ensemble learning to address such issues. Then, the development of a new explainable rule-based ensemble learner, called ReG-Rules, with 5 components is presented. The chapter first, identifies G-Rules-IQR as the suitable learning algorithm (inducer) for the base classifiers induction of the proposed ensemble. Then, it illustrates the three novel methods incorporated into ReG-Rules learner construction. (i) Ranking-based method, which is used to rank the base models before selecting the top ones. (ii) Local Rule Merging method, which represents a post-processing of the induced rules to improve their quality locally and independently. (iii) Combination method based on weighted voting strategy, which is utilised by ReG-Rules learner to build committees of rules to decide its final predictions. Then, several experimental studies are conducted to evaluate empirically and qualitatively ReG-Rules and the aforementioned integrated methods.

Chapter 6. Demonstrates a number of improvements that can be made to a rule-based ensemble system (such as ReG-Rules) to avoid some potential obstacles related to expressiveness, computations and memory resources especially in the

domain of large datasets. Then, solutions implemented resulting in developing a significant extension of the ensemble ReG-Rules learner, which is called *'Consolidated Rules Construction (CRC)'* system. This CRC can compress multiple base classifiers' rule sets into one global expressive rule set. The chapter describes the 5-stages construction of CRC, which have some similarities with ReG-Rules in the first three components. However, in stage 4, a new rules consolidation method is integrated in CRC to derive the global rule set that will be used directly in predictions (stage 5). This is followed by an empirical evaluation that aims to evaluate the performance of CRC.

Chapter 7. Concludes the thesis with a summary of the presented research. It highlights the contributions to knowledge and the extent to which the project aims and objectives have been met is examined. Also, the chapter lists the publications that have been produced during the project and outlines some potential areas for future research by which this work can be extended.

1.7 Summary

This chapter has discussed the ensemble learning strategy, which is often used to improve the accuracy of the classification model. However, this often goes at the expense of expressiveness and explainability of the predictive model learned. The chapter emphasizes the need for an expressive method that can be integrated in a predictive ensemble leaner without compromise made to its accuracy. It has also described the need to develop an appropriate classification algorithm that can be used to induce the suitable base learners of the ensemble. The chapter also introduced the methodology adopted in this research including the evaluation procedures used to measure the experimental studies. The contributions to knowledge and publications accomplished during the project have been listed with brief description. Finally, the structure of the thesis is outlined at the end of this chapter.

Chapter 2

Background

This chapter introduces essential background of predictive data mining algorithms and their limitations in the classification tasks for two paradigms: single-based and ensembles. Classification rule induction and the algorithms related in its two types: 'divide and conquer' and 'separate and conquer' are reviewed in more depth. Also, the chapter explores several characteristics of ensemble methods, and then describes a number of criteria for evaluating the predictive performance of ensemble learning systems.

2.1 Introduction

As previously discussed in Chapter 1, instead of sacrificing accuracy on the training set in order to avoid overfitting, utilising predictive ensemble learning approaches can be an ideal way to reduce the problem of overfitting while improving the accuracy of the model. This is the main advantage of using ensemble methods according to several studies found in the literature such as [1, 3, 5, 8–10, 24, 25].

However, the lack of interpretability is a major drawback in most ensemble techniques especially that the importance of developing a interpretable predictive learning model is not less than obtaining a very accurate one [2] in many critical applications such as medical diagnoses, financial analysis, terrorism detection, etc. In other words, the benefit to use most existing ensemble systems is a choice of a black box model prioritizing accuracy.

The aim of this research is to develop a predictive learner that can be interpretable by humans (expressive) while retaining the key advantage of ensembles, which is the better accuracy.

Among the set of objectives established in Chapter 1 that to be accomplished to meet the research's aim is the following:

> *"Critically assess predictive learning techniques of rule-based and ensemble systems and their limitations."*

Accordingly, this chapter thoroughly reviews the literature around predictive learning systems, in which they can be broken down into two categories:

1. Single learning systems.
2. Ensemble learning systems.

With regards to (1), the chapter provides in Section 2.2 an overview of several predictive data mining algorithms. This involves investigating their suitability to be used as a single base learning algorithm (inducer) to produce multiple base classifiers for the proposed ensemble systems. However, approaches to classification rule induction will be surveyed in more depth because they are much closer to the white box models than other techniques as previously discussed in Chapter 1 (see Figure 1.1). Rule induction strategies will be divided in Section 2.3 into two categories: (a) *'divide and conquer approaches'*, [1] which can extract classification rules form decision trees, (b) *'separate and conquer approaches'*, which can generate IF-THEN rules directly from training datasets. Each strategy is examined in this chapter with more detailed descriptions of its popular algorithms. However, the modular rule induction approaches that this project focuses on are based on separate and conquer strategy. Therefore, the algorithms based on this strategy will be reviewed in depth in Section 2.4.

Regarding (2), the ensemble learning system, the general concept and the philosophy of this system are illustrated in Section 2.5. Also, the relevant literature on categorising ensemble methods in classification tasks is comprehensibly discussed in Section 2.6. These characteristics of the ensemble methods include the following four factors: inter-classifiers relationship (Section 2.6.1), diversity generator (Section 2.6.2), combining method (Section 2.6.3), and ensemble selection (Section 2.6.4).

Furthermore, evaluating the performance of an ensemble is very important for assessing the quality of the ensemble and regulating its parameters accordingly. Several criteria such as computational complexities, interpretability of the resulting ensemble, scalability to large datasets, and robustness are reviewed in Section 2.7. Finally, a short summary will be provided in the last section (2.8).

[1] Divide and conquer approaches are also known as **T**op **D**own **I**nduction of **D**ecision **T**ree (TDIDT) techniques and the two terms are used interchangeably in this thesis.

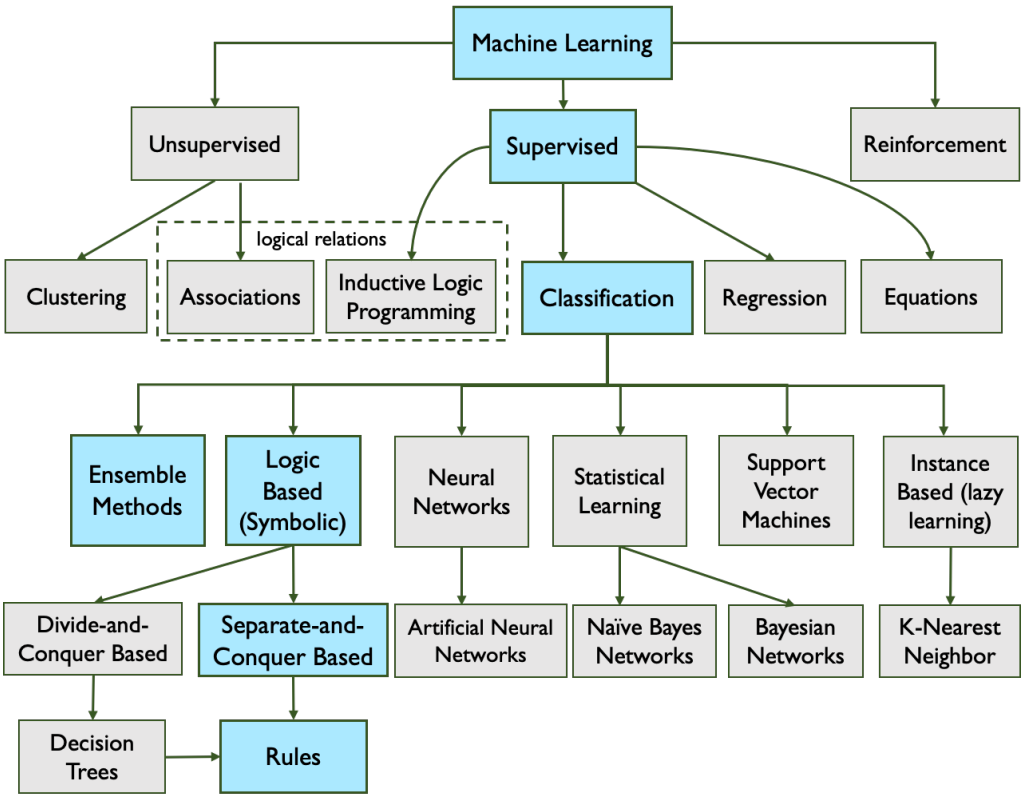


Figure 2.1: Machine Learning Taxonomy, blue colour refers to the approaches related to the work presented in this thesis

2.2 Overview of Predictive Data Mining Algorithms

Predictive data mining are typically supervised methods. They are used to train models from class-labelled data. Then, each trained model (classifier) can be used to predict the class label of previously unseen data instances.

The machine learning taxonomy presented in Figure 2.1 demonstrates a number of popular learning methods such as Decision Trees, Classification Rules, Support Vector Machines (SVM), k-Nearest Neighbors (kNN), and Artificial Neural Network (ANN). The main characteristics and limitations of these methods will be explained in this section. However, rule induction algorithms, which is related to the methodology adopted in the research presented in this thesis will be more closely investigated further in Section 2.3.

2.2.1 Decision Trees Approach

Top Down Induction of Decision Trees is a well-known technique in data mining community, which originally began in the 1970s due to the need for discovering survey data and statistical programs [26]. Since then, a considerable amount of literature has been published discussing decision trees. A traditional decision tree [27] uses a top-down recursive divide-and-conquer strategy to construct a

classifier from a training dataset. As it can be seen in Figure 2.2, decision tree has a tree-like structure which consists of a number of *nodes* and *branches* created by a process known as *'splitting on the value of attributes'* [5]. The partitioning is represented by a sequence of tests. Each internal node symbolises one test of an attribute value, and the branches from the node are labelled with the possible outcomes of the test (classifications). Usually, the test compares an attribute value with a constant value. The splitting process continues until each branch can be labelled with just one classification, i.e. leaf nodes are reached [3,5,28]. However, selecting the best attributes to split the data affects the class distribution of the following branches which is a difficult task in tree inductions [1,26,27,29].

On the other hand, one of the main advantages of tree-based models is their simplicity to be converted to a set of rules by easily transforming each leaf in the tree into a rule [1, 2]. Thus, a rule represents a direct path from root to a leaf node. This level of simplicity is what makes decision tree classifiers so popular [1, 29, 30]. However, the process suffers from several limitations, which will be discussed further in Section 2.3.

As previously explained in Chapter 1, this research uses rule-based methods to generate a base learner for an ensemble because they are much closer to the white box models than other popular data mining techniques. Decision trees can be a source of rules for a classifier. However, the suitability of this type of rule representation as a base learner for an expressive ensemble model must be investigated first according to the second objective of this thesis.[2] Therefore, the decision tree construction strategy will be further reviewed in Section 2.3.1.

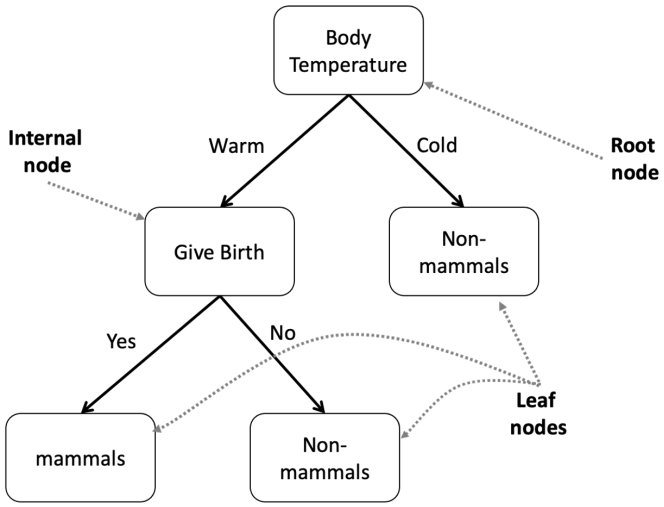


Figure 2.2: A decision tree for the mammal classification problem (source: adapted from [2])

[2] The second objective of this project as stated in Chapter 1 is to measure and compare the expressiveness of rule based models and develop an appropriate rule-based predictive algorithm suitable as base learner for an ensemble.

2.2.2 Rule Induction Approach

As mentioned in Chapter 1, the rule-based predictive methods use a set of IF-THEN rules for classification. The *left-hand side* (LHS) or precondition of a rule is a series of test (logically conjunctive together) just like the internal nodes in decision trees. The *right-hand side* (RHS) or consequent contains a class prediction just like the leaf nodes in decision trees [2]. If an instance satisfy all the conditions of the LHS of a rule, then we can say that the instance is covered by that rule. The two common strategies to generate classification rules are the *'Divide-and-Conquer'* and the *'Separate-and-Conquer'* approaches. The former is based on decision trees by simply create one rule for each path in the tree from the root to a leaf node. The latter approach induces modular predictive rules directly from the training data without having to construct a decision tree. Both ways have advantages and disadvantages and since this thesis focuses mainly on predictive rule-based algorithms, these two types of rule induction will be more thoroughly investigated in Section 2.3.

2.2.3 Support Vector Machine (SVM) Approach

Support Vector Machines (SVMs) are a supervised machine learning technique presented in 1992 by Vladimir Vapnik in [31] and are used for regression analysis and classification. It has received considerable attention in the machine learning literature [32]. This technique is based on statistical learning theory and ideally works well on binary classification problems. The main aspect of SVM approach is its ability to avoid the curse of dimensionality by transforming the training data into a higher dimension. Within this new dimension, SVM approach finds the optimal linear *'separating hyperplane'* (decision boundary) to separate two classes. The basic idea of linear SVMs is conceptually illustrated in Figure 2.3.

The idea is to maximise the *margin*, i.e. the distance between the hyperplane and the closest instances of the two classes. Those closest instances, which called *'support vectors'* are used to define the margin that separates the two classes [1, 32, 33]. Although transforming data attributes into a higher dimension seems a promising approach with regard to the accuracy of SVM models, it often requires costly computations.

In addition, when the data is not linearly separable, SVM needs a non-linear separating hyperplane. One potential solution is to transform the data into an infinite dimensional space, but this is an even more computationally expensive task and may suffer from the curse of dimensionality problem [32]. Therefore, SVM uses *'kernel tricks'* method to construct non-linear separating surfaces where the learning process is performed on the original data attributes and not on the trans-

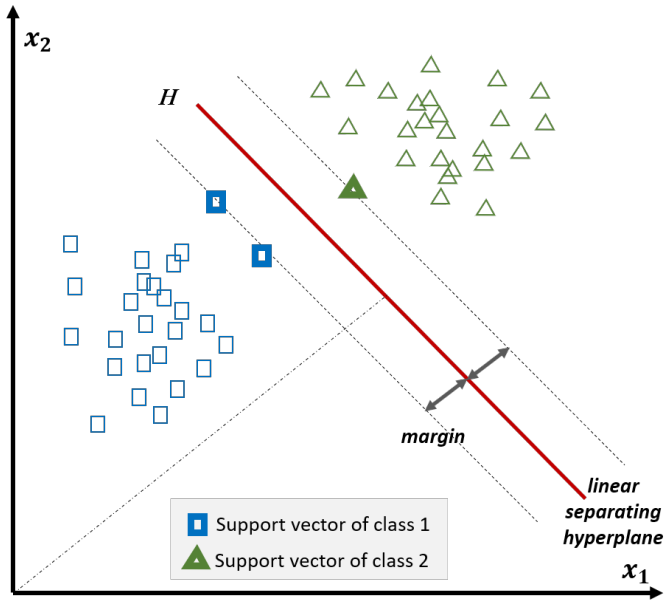


Figure 2.3: Basic idea of linear support vector machines (source: adapted from [33])

formed attributes. SVM is often applied in applications such as handwriting digit recognition, object recognition, and speaker identification. However, selecting a 'good' kernel function is not an easy task and often increases the training time considerably. Also, the user of SVM approach has to define a set of hyper-parameters such as the type of kernel functions to use and the cost function C and gamma for introducing a binary variable for each of the attribute values [32].

Because of these overheads and model complexity, especially when dealing with large datasets, SVM's classifiers are difficult to understand and are referred to as black box models. In fact, in many applications it's almost impossible to explain the logic behind model predictions or visualise their impacts [28, 34]. Therefore, SVM is not a suitable approach for critical applications where the interpretability of the model is highly desirable [35].

2.2.4 Instance Based Learning Approach

It is a statistical method known as *lazy-learning* approach, as it does not require explicit learning phase. One of the most popular instance-based learning techniques is the **k-Nearest Neighbors (kNN)** algorithm. It was first presented in the early 1950s by Fix and Hodges in [36] and its main idea based on the principle that, most of the instances that exist close to each other within a dataset are sharing similar properties [1, 29, 34]. According to this principle, the learning process in kNN classifiers is performed by comparing a new testing instance with all the training instances that are similar to it. Then, the unlabelled new instance will be classified based on the class of its neighbours. In case of having more than one la-

bel close to this unclassified instance, the new instance will be classified according to the majority class of its neighbours [32]. The proportional distance between instances is defined by a *distance metric* such as Euclidean distance, which typically tends to minimise the distance between the instances belong to the same class and maximise the distances between instances of different classes.

Despite the very simple implementation of kNN algorithm and the little number of parameters required (distance metric and k), this approach have several limitations, such as the low efficiency being a lazy learning method and the heavy dependency on the value of k, etc. [37]. The authors of [29] argue that kNN algorithm suffer from the lack of principled strategy to select k except through cross-validation or similar computationally-expensive technique. This besides the high cost of classifying new instances due to the absence of a learned model. According to [37], every computational single new instance needs $O(n^2)$ to be classified. The effect of the choice of k on the performance of kNN classifiers is described in Figure 2.4. Due to this high a drawback, which is likely to be even more expensive in large datasets, kNN seems to be not suitable as a base inducer of the ensembles proposed in this project. The reader is referred to [29,37] for a detailed review of kNN algorithms and instance-based learning approach in general.

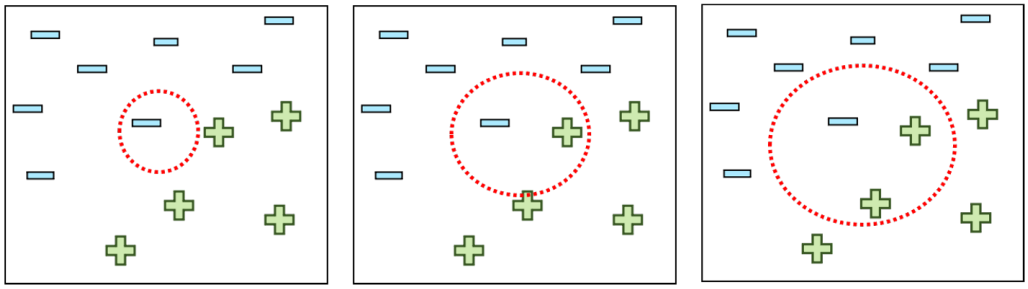


Figure 2.4: The 1-, 2-, and 3-nearest neighbour (source: adapted from [32])

2.2.5 Artificial Neural Network (ANN) Approach

The study of artificial neural networks (ANN) was inspired by efforts to mimic biological neural systems. It is one of the most popular data mining techniques. In [38], Donald Hebb introduced the notion of a simple rule for adjusting the intensity of connections between biological neurons in 1949. Such a rule requires only local information, and this type of locality was the real motivation in the literature for developing approaches in neural network learning. The first important advance in the development of ANN algorithms was in 1957 when Rosenblatt presented the first neural model [39], and later introduced the tow-layered perception in 1962. However, a typical neural network that often used for classification problems was presented by Hecht-Nielsen in [40], and called *'multi-layered*

feed-forward neural network'. As shown in Figure 2.5, it consists of a collection of connected neurons organised in a layered cascade as follows: (1) an input layer (corresponding to attributes); (2) one or more hidden layers; (3) an output layer (corresponding to classes). Signals travel from the input layer to the output layer, passing through the hidden layer(s) via connections. Each connection has a weight associated with it. The network model learns by modifying the weights in order to be able to predict the correct class label for a certain instance in the input layer. The reader referred to [41], a textbook by Haykin, for a thorough overview of both supervised and unsupervised neural networks.

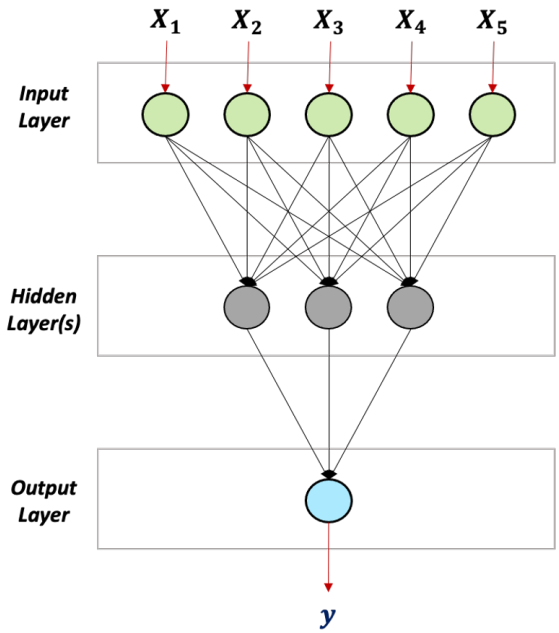


Figure 2.5: Example of a multi-layer feed-forward ANN (source: adapted from [32])

The advantages of artificial neural networks are: (1) their high predictive accuracy including their ability to deal with noisy data; (2) their practicality of using with little information about the relationships between attributes and classes; (3) their suitability for continuous attributes; (4) their potential to be parallelised to speed up the computation process. According to [1], these factors make neural network based algorithms more popular. However, ANN suffers from several drawbacks that can be summarised as follows: (1) a high computational cost and therefore its suitability is limited to the application where this is feasible; (2) requiring a number of parameters that often should be determined empirically; (3) the extreme difficulty to interpret or understand the reason behind model predictions and consequently mostly serve as black-box models. These shortcomings made neural network less desirable for decision makers [1] and therefore not suitable as a base inducer of the ensembles proposed in this project.

2.3 Rule Induction Strategies

Approaches to classification rule induction are surveyed more thoroughly in this part of the chapter. The rule-based algorithms that are described here can be divided into the following: (1) **T**op **D**own **I**nduction of **D**ecision **T**ree algorithms (TDIDT), also known as '*divide and conquer*' approach, which will be discussed in the next section. Please note that both terms are used interchangeably in this thesis. (2) 'Covering' algorithms, also known as '*separate and conquer*' approach which will be discussed in Section 2.3.2.

2.3.1 Divide and Conquer Strategy

As mentioned in Section 2.2.1, generating classification rules from an intermediate form of decision tree using divide and conquer approach is a widely used technique. However, these rules are usually pruned to remove the irrelevant and redundant conditions (terms), which is considered a necessary but unfavourable outcome of tree representation of rules. In other words, the rule generation process using divide and conquer strategy requires the following three steps:

Step 1. Decision Tree Construction: constructing a decision tree first is a necessary and unavoidable step before the conversion to an equivalent rule set taking place. In this section, some common decision tree algorithms will be reviewed in order to investigate their suitability to meet the objectives of this research.

One of the common decision tree algorithm is ID3 (Iterative Dichotomiser 3), which is based on a prior work on 'concept learning system' established by [42]. The main reason for ID3 popularity lay in its information formula, which is used to select the appropriate attribute that has the highest information gain. However, ID3 has some disadvantages, for instance: (1) in feature selection procedures, ID3 biases toward attribute that have more values despite the fact that multi-values attribute is not always the best choice; (2) multiple logarithmetic operations are required in the process of attribute selection, i.e. high computational cost; (3) It is hard to control the tree size. Therefore, several techniques were suggested later with some degree of ID3-compatibility in an effort to develop a more efficient ID3-based algorithm; for example, ID4, ID5, ID5R, and IDL algorithms.

Regardless of whether these ID3 evolutions were successfully handled the original ID3 problems or not, C4.5 algorithm, which is also introduced by Quinlan is considered to be more effective and became a cornerstone of the newer data mining algorithms to compare with [1]. This widely used algorithm utilises information-theoretic entropy as a purity measure for attribute selection process

[3]. The attribute that would reduce the entropy the most by splitting the data is used to expand the tree. As a result, the attributes near the root have a stronger influence on the class label than the ones in the lower part of the tree, and hence the reliability of the selected attributes reduces with the growth of the tree [3]. This also, would likely increase the size of the tree and consequently increase the risk of overfitting, especially that tree models are often prone to overfit the training data.

Step 2. Extracting Rules from Decision Trees: Each branch of the tree corresponds to a classification rule, and so extracting rules from it can be accomplished by converting each direct path from root to leaf nodes into a rule. Although, it is easy to perform the transformation, this procedure might produce rules that are far more complex than necessary, and they are often executed in irrelevant order [2]. In fact, they may inherit all the complexity of the source or might become more difficult to understand than the corresponding trees in some cases [1].

Furthermore, the redundancy problem that is mentioned at the beginning of this section leads to so called '*replicated subtree problem*', which has been recognised by Cendrowska in [43] as the main reason for overfitting in decision trees. Although, Cendrowska never uses the term replicated subtree, her study comprehensively described this popular drawback very well. She criticised the principle of deriving decision rules from trees in comparison with generating modular rules directly from training data. Cendrowska argues that decision trees cannot easily represent the implicit disjunction among the different rules. Thus, forcing rules with no common attributes to fit in a tree structure would require adding unnecessary, meaningless rule-terms.

To explain how this problem happens in decision tree representation of rules, consider the following rule set 2.1 as an example:

$$\left.\begin{array}{l} \text{IF } A_1 \text{ AND } B_2 \text{ THEN Class } = x \\ \text{IF } C_2 \text{ AND } D_3 \text{ THEN Class } = x \\ \text{Otherwise} \rightarrow \text{ Class } = y \end{array}\right\} \text{ Rule set} \tag{2.1}$$

The complexity of the corresponding tree depends on the number of attribute values that are possible to be selected for splitting. Let the four attributes (A, B, C, D) each have only three possible values $(1, 2, 3)$ and let attribute A be selected at the root node. Thus, the simplest form of the tree to represent the above rules is demonstrated in Figure 2.6.

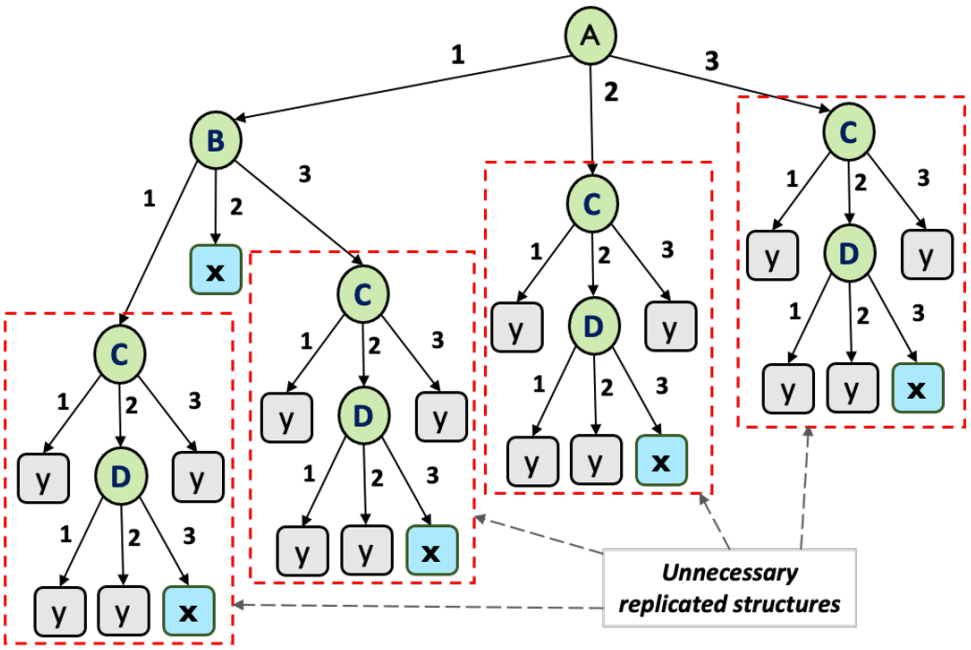


Figure 2.6: An example of a replicated subtree problem

Consequently, by following divide and conquer approach, the total of 21 rules will be derived from the decision tree that is shown in Figure 2.6. The rules that predict class x only is shown below in rule set 2.2 :

$$\left.\begin{array}{l}\text{IF } A_1 \text{ AND } B_1 \text{ AND } C_2 \text{ AND } D_3 \text{ THEN Class } = x \\ \text{IF } A_1 \text{ AND } B_2 \text{ THEN Class } = x \\ \text{IF } A_1 \text{ AND } B_3 \text{ AND } C_2 \text{ AND } D_3 \text{ THEN Class } = x \\ \text{IF } A_2 \text{ AND } C_2 \text{ AND } D_3 \text{ THEN Class } = x \\ \text{IF } A_3 \text{ AND } C_2 \text{ AND } D_3 \text{ THEN Class } = x\end{array}\right\} \text{ Rule set} \qquad (2.2)$$

Step 3. Rule Post-pruning: The mechanism that often used to prune divide and conquer rules is similar to a tree pruning method, which is based on the error rate estimation at each internal node in the tree [2]. For each condition (term) in a given particular rule, work out the instances covered by the rule before and after removing the term, and then, calculate the error rate of the rule in both cases. If the rule is better without the current condition, remove it and continue investigating another condition in the same rule. The rule will be completed when there is no further improvement can be introduced to it. After pruning all the derived rules, deletion of any duplicated rules is also another important step.

However, there is no guarantee that simplifying rule set this way would produce the best version of the rules [2]. Another solution to find the correct set of conditions is by using optimization techniques, such as genetic algorithms [1, 2].

The main problem of these greedy pruning methods is computational cost. For every condition that is subject to be removed from a particular rule, the whole rule must be re-evaluated on all the training instances. This is usually an expensive procedure, and its cost might be significantly increased for classification tasks that involve a lot of noise in the data [1, 2]. The algorithms C4.5 and its successor C5.0 are typically using these pruning methods to successfully derive a good rule set from a transitional form of decision tree [44, 45]. Note that in the literature, the term C4.5RULES is used to refer to generating rules from a pruned C4.5 tree [1, 2, 46]. However, in addition to the undesirable computational cost, the pruning procedures of these algorithms might needlessly increase the size of the trees without improving the performance of the classifiers.

Another popular algorithm that uses pruning methods to generate rules from decision trees is CART (**C**lassification **A**nd **R**egression **T**ree), which is described by Breiman in [47]. CART tree learning system tries to regulate the tree size and avoid the unnecessary growing of the tree by using a more conservative pruning mechanism called 'cost-complexity optimisation'. According to empirical investigation conducted by Oates and Jensen in [48], CART system often generates a smaller and more accurate tree than C4.5 rules method. However, this tree takes much longer training time to be produced, and it is even more difficult to understand. The reader is also referred to [30,49] for additional descriptions of common divide and conquer rules algorithms.

To sum up, extracting classification rules from the decision trees including the aforementioned widely used models (C4.5, C5.0, and CART) have several limitations that can be summarised as follows: (1) expensive computational cost in the pruning methods of C4.5 and C5.0, which might needlessly increase the size of trees; (2) long training time in CART system which produce rules that might become more difficult to understand than the corresponding trees in some cases; (3) tree models in general are often prone to overfit the training data and thus produce rules with irrelevant and redundant conditions (terms). Therefore, the tree representation of rules is not suitable as a base inducer of the ensembles proposed in this project. Further explanations will be provided in Section 2.4.5.

2.3.2 Separate and Conquer Strategy

The previous section outlined how classification rules are derived from decision trees. Despite the advantage of being easily obtained, and to some extent, understandable by analysts, several drawbacks of this type of rule representation were also highlighted. However, the alternative to divide and conquer approaches have been originally developed by Michalski in 1969 under the name of *covering* strat-

egy [50]. Then, Pagelo and Haussler in 1990 refer as *'separate-and-conquer'* [51] to covering approach. A considerable amount of studies in the literature, such as [1–3, 43, 52] use both terms to describe the same strategy. This type of rule learning can extract IF-THEN rules directly from the training data, i.e., without having to construct a decision tree first, and thus being less susceptible to the potential issues stated in the previous section (2.3.1).

A basic pseudocode for separate and conquer strategy is shown in Algorithm 1. Rules are produced for a given class at a time. After a rule is generated, all instances that are covered by that rule are removed from the current training dataset (the separate part), and the next rule is induced using the remaining training instances (the conquer part) until a terminating condition is met. The terminating conditions (also known as stopping criteria) vary from algorithm to algorithm. However, the common ones are: no more training instances are left; the quality of a rule induced is below the user-specified threshold; or all the remaining instances belong to the same class (pure examples) [1, 2, 52].

Algorithm 1: Basic Separate and Conquer algorithm

```
1  Rule-set = { } ;          //initialise set of rules
2  foreach class C do
3  |   Rule = { } ;          //generate an empty rule
4  |   while terminating condition not satisified do
5  |   |   Rule = Learn.Rule ;
6  |   |   Remove all instances covered by Rule from training data;
7  |   end
8  |   Rule-set = Rule-set + Rule;          //add new completed rule to rule set
9  end
10 return Rule-set;
```

The most important step in the above algorithm is the *'Learn.Rule'* function (line 5), which begins with an empty rule and then gradually appends the best attribute-value pair (rule-term) for the current class C. All the possible combinations of rule-terms should be considered before deciding the best rule-term, which makes this greedy process computationally very expensive. Therefore, different covering algorithms adopt different *search* strategies to reduce the search space of the *Learn.Rule* procedure, while maintaining the quality of the rule. All these strategies use the *general-to-specific manner* to induce the rules [1, 2, 52].

2.4 Separate and Conquer based Algorithms

There are many separate and conquer algorithms. Popular approaches include AQ family of algorithms [50, 53, 54], CN2 algorithm [55], RIPPER [56], and PRISM algorithm [43] and its successors. The first three algorithms will be explored in Sections (2.4.1, 2.4.2, and 2.4.3), respectively. However, PRISM algorithm will be thoroughly investigated in Section 2.4.4 and compared with aforementioned rule induction strategies in Section 2.4.5 because of its relation to several parts of this research.

2.4.1 AQ Family of Algorithms

The original version of the **A**lgorithm **Q**uasi-optimal (AQ) can be considered as the original covering algorithm, which was developed by Michalski in [50]. Various algorithms based on this have appeared in the literature over the span of decades, such as [54, 57–60]. These subsequent tailored implementations of AQ aimed to handle different complicated learning problems. Generally speaking, the variations between the members of the AQ family of algorithms are based on applying different parameter settings. Although, these parameters are not considerably difficult to learn, they might be to, some extent, a possible cause of limiting the usage of the AQ approach.

The basic AQ algorithm is an irreversible top-down search that induces a rule for each class in turn. It starts by selecting one positive example (called *seed example*) and then repeatedly generates all the possible conjunctions of tests (called *complexes*) that cover the seed but do not cover any random selected negative examples. By using rule learning heuristics, all these potential complexes are evaluated before selecting the best complex. The algorithm aims to learn perfect rules that cover all positive examples and exclude all the negatives.

The AQ method has advantages and disadvantages in comparison with other rule-based algorithms such as C4.5RULES [44, 45]. Some of the original disadvantages have been overcome over the years, with additional processes utilised for optimization purposes. One of the main advantages of AQ approach is its ability to generate readable and maintainable rules, like most of the separate and conquer based algorithms. AQ could be further improved to generate good rules that consider only positive examples. However, AQ is computationally expensive as it is considerably slower in comparison with C4.5RULES algorithm. This is due to the main aspect of the learning process of the AQ algorithm, which compares each positive event in the training dataset with all the negatives. However, this aspect can be also considered as a strong point at the same time, since it guaran-

tees that all the decisions to select the best rule-term (complex) are made with the maximum amount of information available [61].

2.4.2 CN2 Algorithm

CN2 algorithm [55] is named after the initials of its designers, **C**lark and **N**iblett. The algorithm integrates the ideas from AQ and ID3 approaches, i.e., induces rules directly from training data using beam search strategy like AQ but with ID3 capability of handling noisy data. The essential observation in rules learning is that a single rule represents a branch in a decision tree. Thus, the first version of CN2 uses ID3's information gain (entropy) to evaluate each possible condition instead of only concentrating on conditions that randomly partition pairs of positive and negative examples like in AQ algorithm. Using this search heuristic to select the best condition makes CN2 less susceptible to a potential problem that AQ algorithm would suffer from, which is selecting a mislabelled negative instance as a positive seed example, and thus be forced to induce a rule that covers this example. This is a great advantage over AQ algorithm, which makes CN2 the first rule learning system that addressed the overfitting problem [3].

Additionally, the original CN2 adopts the ID3 algorithm's method to deal with multiple classes by selecting the majority class among the instances that covered by the complete rule [3]. However, a study in [62] later optimised CN2 and suggested different search heuristic that can deal with each class in the training data as a separate concept, thereby considering all the instances that belong to the current class as the positive examples and all the rest as the negatives. This approach, which they called *unordered*, has become very popular and is also known as *one-against-all*. Multiple successors for CN2 algorithm have been found in the literature, including *m*FOIL [63], ICL [64] and BEXA algorithms [65]. The reader is referred to [3] for a brief review of these methods.

2.4.3 RIPPER Algorithm

The RIPPER (**R**epeated **I**ncremental **P**runing to **P**roduce **E**rror **R**eduction) algorithm was introduced by W.Cohn in [56]. It is based on the idea of incremental reduced-error pruning (IREP) presented by Fürnkranz and Widmer in [66]. RIPPER algorithm have addressed the overfitting problem efficiently [3]. The algorithm can be understood in a three-step process: (1) *grow*, (2) *prune*, (3) *optimise*.

For each individual rule for a given class, the training data is split into two sets (growing set) and (pruning set). In the first step, the rule will be specialised by adding new terms using the growing set, and then the information gain criterion

is utilised to determine the next term. When adding a new term to the rule no longer reduces entropy, the specialisation step stops and the rule will be generalised by removing unnecessary terms using the pruning set. Steps one and two are repeated until the general stopping criterion of the rule set is reached. At this point, a global optimisation strategy is applied to the rule set using a diversity heuristic. In case of finding no more rules to learn, a default rule (with empty RHS[3]) is added for the most frequent class.

According to [3], RIPPER provided a powerful rule based model that was used for several practical applications. However, the algorithm is not as accurate as tree based models when compared with C4.5RULES approach [44, 45], and also using one class as a default prediction has some disadvantages. Several studies have tried to improve RIPPER approach; for example, an alternative variation of the algorithm named 'JRIP' is implemented in WEKA [2], and considered to be among the most competitive rule learning systems available today.

Another interesting approach, which involves incorporating some ideas from fuzzy rule induction into RIPPER, is called Fuzzy Unordered Rule Induction Algorithm (FURIA). The approach was proposed in [67] and the experimental results revealed that FURIA as a hybrid algorithm outperformed RIPPER and C4.5RULES in terms of classification accuracy. However, according to [68], rules created by FURIA models might not always be as human-readable. Also, a main disadvantage of RIPPER algorithm and its predecessors is that certain important information in the training data might be prevented from being utilised during the rules' induction step because some instances are split into the pruning set. Also, for a similar reason, some incorrect rules might be kept during the pruning step as a result of limited information in the pruning set to detect the error.

2.4.4 Modular Rule Induction using PRISM Algorithm

As mentioned in Section 2.3, 'Modular Rules' refers to the representation of rules that often can not be fit into a decision tree without creating at least one irrelevant feature [5]. This type of limitation of tree representation of rules was explained previously in Section 2.3.1 using the example of replicated subtree problem that can be seen in Figure 2.6. The problem that first recognised by Cendrowska in [43] and considered to be the main cause of the overfitting problem. In fact, she criticised the whole principle of building a decision tree in order to derive decision rules from it. Therefore, she developed PRISM [43] as an alternative expressive rule induction approach, which will be reviewed in this section.

[3]RHS refers to the right-hand-side of the rule, also known as 'rule consequent'

The Basic PRISM Algorithm:

The PRISM algorithm is based on 'separate and conquer' strategy and developed with the aim of inducing modular classification rules directly from training data in the form of a 'IF-THEN' rule set. As described in Algorithm 2, the basic PRISM approach can handle only categorical attributes. The outer loop iterates over the classes, starting with a target class and generating only correct and 'perfect' rules by inducing one rule at a time for the current class. Then, PRISM measures the quality of an attribute-value pair $\alpha = v$ (rule-term) by calculating its conditional probability $\mathbb{P}(C_i|\alpha = v)$ for the target class C_i. Hence, each rule is specialised term-by-term by selecting the term that maximises the conditional probability of the rule's selected target class. The training stops once the rule only covers instances belonging to that pre-assigned target class. These instances covered by the induced rule will be removed from the training data before the induction of the next rule commences. The process is repeated until there are no instances left in the training data that match the target class. Then, the training set will be reinitialised to its initial state before the same procedure is carried out for the next target class.

Algorithm 2: Cendrowska's original PRISM Algorithm

1 **for** *each class* C_i **do**
2 | **Step 1:** Calculate the probability of occurrence, $\mathbb{P}(C_i|\alpha = v)$, of the class C_i for each attribute-value pair $\alpha = v$ (rule-term t_α) ;
3 | **Step 2:** Select the pair $\alpha = v$ with the largest $\mathbb{P}(C_i|\alpha = v)$ and create a subset of the training set comprising all the instances that match the selected $\alpha = v$;
4 | **Step 3:** Repeat step 1 and step 2 for this subset until a subset is reached that contains only instances belong to class C_i. The induced rule R is then the conjunction of all the selected $\alpha = v$ pairs (rule-terms) ;
5 | **Step 4:** Remove all instances covered by R from the training set ;
6 | **Step 5:** Repeat steps 1 to 4 untill all instances of class C_i have been removed from the training set.
7 **end**

Further Variations of the PRISM Algorithm:

Cendrowska's PRISM does not indicate any method of dealing with the following:

1. Tie-breaking problem, i.e. two or more rule-terms have the same conditional probability.
2. Clashes in the training data, i.e. two or more instances are identical but have different class labels.
3. Conflict resolution in the testing stage, i.e. two or more rules with different class labels but covered the same example.
4. Continuous attributes, i.e. only categorical features are considered.

Therefore, PRISM sparked work on a range of different PRISM variations with the aim of improving the algorithm, also known as the ***'PRISM family of algorithms'***. One of the early attempts to improve the performance of original PRISM was introduced by Bramer in [69], namely ***N-Prism*** *algorithm*, which incorporates some additional features to overcome the aforementioned shortcomings. These revised features will be investigated more closely in the next chapter as they are, to some extent, related to the algorithms developed in this thesis, and also they have been applied to the whole PRISM family. N-Prism was implemented in the Barmer's Inducer software [70,71]. A number of experiments were conducted to compare this extended version of PRISM with the divide and conquer rule induction approach[4]. The results of these experiments will be explored in the next section, where comparisons between PRISM family of algorithms and the other rule based algorithms are detailed.

More variations of PRISM exist such as ***PrismTC*** and ***PrismTCS*** algorithms, where *'TC'* refers to Target Class and *'S'* indicates that class with Smallest number of instances is considered first [69, 71]. PrismTC always uses the majority class and PrismTCS uses the minority class. The main difference between these two variants of PRISM and the basic PRISM is that PrismTC and PrismTCS introduce an order in which the rules are induced, where there is none in the basic PRISM approach. Another difference is that the training set in the original version has to restored to its full original size for each of the classes, while in PrismTC and PrismTCS the full training set only needs to be processed once no matter the number of classes. Therefore, both variations can remove the outermost loop in the original PRISM (line 1 in Algorithm 2). This aspect can be considered as a

[4]'Divide and conquer' rule induction is also known as TDIDT based rules; and both terms are used interchangeably in this thesis.

computational advantage and also a useful aid to the development of a distributed version of PrismTCS, which is another member in the PRISM family called **PMCRI** (Parallel Modular Classification Rule Induction) [72]. PMCRI allows to parallelise any member of the PRISM family of algorithms.

Generally speaking, existing literature shows that PRISM's successors have focused mainly on improving the algorithm's predictive accuracy and scalability. However, to the author's best knowledge, there is no PRISM based algorithm that can produce numeric rules from batch data in an expressive and computationally efficient way. The current implemented method of processing continuous attributes in PRISM family of algorithms is based on a local discretisation method called cut-point calculations (also known as binary splitting), which produces rule-terms of the form ($\alpha < x$ and $\alpha \geq y$) or ($\alpha \leq x$ and $\alpha > y$) where x and y are two current values of the attribute named α. This way of handling numeric values is computationally expensive, and hence results in long processing times. Therefore, a new proposed heuristic approach will be introduced and experimentally evaluated in the next chapter. The approach introduces a new rule-term structure in the form of ($x \leq \alpha < y$) instead of two separate rule-term combinations, which greatly enhances the readability of the individual rules[5].

2.4.5 Comparing PRISM with other Rule Induction Algorithms

In this section, PRISM approach is compared with the 'divide and conquer' rule induction approach (TDIDT based rules), and then with other popular 'separate and conquer' rule induction approaches (i.e. AQ, CN2, and RIPPER).

PRISM Approach vs. Divide and Conquer (TDIDT Rules) Approach

According to [3, 43], the key advantage of PRISM comparing with decision trees is based on its induction strategy. While PRISM follows the theory of overlapping rules, decision trees are constrained to discover a theory with non-overlapping rules. More specifically, PRISM has an information theoretic basis to avoid inducing classification rules that contains redundancy, which is considered a necessary but undesirable feature of decision trees that may result in more complex models in large datasets.

Another difference is that PRISM generally has a preference for leaving a test instance 'unclassified' rather than giving it a wrong classification. In critical domains, this may be an important feature. According to [5, 69], when noise exists, PRISM is more stable and achieves better predictive accuracy than TDIDT even if

[5]This approach is one of the contributions of this research and it was published in [19].

the level of noise is very high in the training data. An additional advantage, a rule produced by PRISM algorithm can be separately evaluated or even removed without affecting the entire set of rules (the constructed model). In contrast, there is no possibility of making any modifications in a built tree without reconstructing the entire tree [1, 3, 5].

Table 2.1: Opticians' decision table for fitting contact lenses. Source: adapted from [43]

Instance id	Attribute value				lenses?	Instance id	Attribute value				lenses?
	A	**B**	**C**	**D**			**A**	**B**	**C**	**D**	
1	1	1	1	1	no	13	2	2	1	1	no
2	1	1	1	2	soft	14	2	2	1	2	soft
3	1	1	2	1	no	15	2	2	2	1	no
4	1	1	2	2	hard	16	2	2	2	2	no
5	1	2	1	1	no	17	3	1	1	1	no
6	1	2	1	2	soft	18	3	1	1	2	no
7	1	2	2	1	no	19	3	1	2	1	no
8	1	2	2	2	hard	20	3	1	2	2	hard
9	2	1	1	1	no	21	3	2	1	1	no
10	2	1	1	2	soft	22	3	2	1	2	soft
11	2	1	2	1	no	23	3	2	2	1	no
12	2	1	2	2	hard	24	3	2	2	2	no

Concerning expressive power, the authors of [43] point out that decision trees are not always appropriate in expert systems. This can be explained using Cendrowska's case study, which was taken from the world of ophthalmic optics. Table 2.1 shows a decision table for an optician for fitting contact lenses where each instance represents a description of a classification in terms of values of four attributes (A, B, C, D). The potential values of the attributes are listed below:

A: the age of the patient

1. young
2. pre-presbyopic
3. presbyopic

B: patient's spectacle prescription

1. myope
2. hypermetrope

C: patient has astigmatism ?

1. no
2. yes

D: patient's tear production rate

1. reduced
2. normal

Accordingly, the optician has to decide whether the patient should be fitted with *hard* or *soft* contact lenses or if there are *no* contact lenses suitable to the patient. Now assume that the patient requiring contact lenses was a presbyopic $(A = 3)$ with high hypermetropia $(B = 2)$ and astigmatism $(C = 2)$. From the decision table, PRISM can directly induce the rule (2.3):

$$\text{IF } A = 3 \text{ AND } B = 2 \text{ AND } C = 2 \text{ THEN decision } \rightarrow \text{ No lenses} \qquad (2.3)$$

Thus, the optician within a couple of seconds can take the decision if no contact lens is suitable for this patient. However, assume that the decision tree in Figure 2.7 was used as the knowledge base for an expert system. It would be unable to make a decision without information about attribute D (the tear production rate). Hence, an unnecessary test of the tear production rate will be advised by the optician requiring a lot of time and also may result in an additional fee to be paid by the patient. The consequences could be even more serious than wasting patient's time and money if attribute D involved surgery.

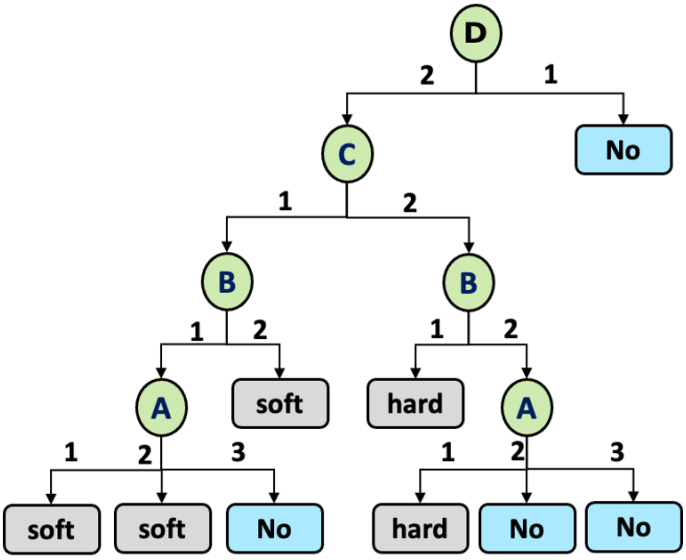


Figure 2.7: Decision Tree. Source: adapted from [43]

Prism Approach vs. other Separate and Conquer Approaches

PRISM algorithm (Section 2.4.4), AQ family (Section 2.4.1), CN2 (Section 2.4.2), and RIPPER (Section 2.4.3) follow a very similar top-down search heuristic that induces rules directly from training data for each class in turn. However, comparing with AQ , PRISM is not controlled by a particular random selected pair of positive and negative examples (seeds), which is an essential learning step in the AQ family. Note that this procedure may cause a potential overfitting problem if AQ algorithm selects a mislabelled negative instance as a positive seed example

and thus forced to induce a rule that covers this example. Also, AQ compares each positive seed in the training data with all the negatives, which is computationally very expensive.

Regarding CN2 algorithm, it uses ID3's information gain (entropy) to select the best pairs of positive and negative instances; thus, it is less susceptible to the problem that AQ suffer from. However, as CN2 combines ideas from AQ and 'divide and conquer' approaches, it is possible to be suffered from the problems that might be existed in one of them. Therefore, PRISM is considered a more stable algorithm comparing with AQ and CN2 because it is not depending on any prior selected seeds in its learning process.

With respect to RIPPER algorithm, splitting the training data into growing set and pruning set makes the classifier not as accurate as tree based models. Some incorrect rules might be generated during the growing step as a result of not having enough information in the set and some other incorrect rules might be kept during the pruning step due to the same reason. Also, unlike PRISM, RIPPER is not abstaining as it uses one class as a default prediction in case of finding no more rules to learn in the growing step.

To sum up, the above comparisons support the benefit of choosing the strategy of PRISM family of algorithms over other rule induction approaches to develop an appropriate rule based predictive algorithm that can be suitable as an inducer for the ensemble systems proposed in this thesis, and thus, achieving objective 2 of the project. Please note that this decision will be further investigated experimentally in Chapter 3. Moreover, some practical and computational issues related to the implemented versions of PRISM family will be also discussed and addressed in Chapters 3 and 4 whereas the rest of this chapter will focus on the literature around predictive ensemble learning.

2.5 Overview of Predictive Ensemble Learning

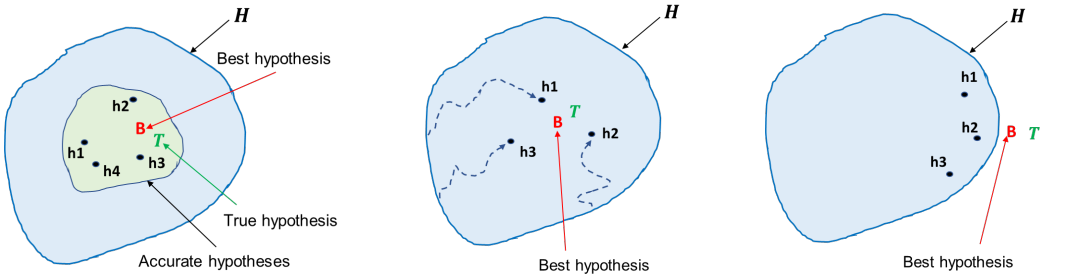
Generally speaking, ensemble methodology stimulates our nature to look for several views before making any critical decision [24]. We mentally assess the individual views and combine them to attain our ultimate choice. Similarly, ensemble learning systems consist of multiple classifiers, each trained on a different subset of data, and produces a single prediction (vote). Combining these votes (decisions) using a some kind of voting approach is likely to create an ensemble with a higher level of overall predictive accuracy than its base learners. The ensemble methodology, referred to as a system of systems and considered to be one of the most effective strategies to improve prediction performance in data mining [73].

It is mostly applicable in all fields where classification algorithms can be ap-

plied. Examples of applications where ensemble have been successfully deployed include: finance [74, 75], manufacturing [76, 77], text categorisation [78, 79], medicine [80, 81], cheminformatics [82, 83], geography [84], information security [85, 86], image retrieval [87, 88], recommendation systems [89, 90]. The next section will illustrate the philosophy of ensemble systems in general and why they usually work better than stand-alone systems.

2.5.1 Philosophy of Ensemble Systems

The goal of any predictive classification model is to obtain good performance. The most common performance measure in predictive data mining algorithms is *accuracy*. Several studies argue that combining the predictions of multiple models learned from the same data can create a more reliable ensemble model with higher level of overall accuracy than its base learners. Kuncheva [91] states that there are three fundamental reasons for this.



(a) **STATISTICAL** (b) **COMPUTATIONAL** (c) **REPRESENTATIONAL**

Figure 2.8: Three essential reasons for using ensembles classifiers. Source: adapted from [91]. The outer curve indicates a space of hypotheses (H), the inner curve denotes the set of hypotheses with good accuracy in the training data, the points labelled (B) is the best hypothesis and (T) is the true hypothesis.

1. Statistical Reason: The aim of a learning algorithm is to find the best hypothesis in a space H of hypotheses (Figure 2.8). If the amount of the training data is too small compared with the size of the space H, a statistical problem arises. Without enough training data, the learning algorithm can find many hypotheses in the space H with a good performance on the training data, but may have different performance in the testing data. Selecting one classifier as the solution for a classification problem, may lead onto the risk of making a bad choice for the problem, such as choosing an overfitted model. This risk is significantly reduced if we construct an ensemble out of these accurate classifiers and the learning algorithm averages their votes. As it can be seen in the Figure 2.8a, the point labelled T is the true hypothesis and by averaging the accurate hypotheses (inner space), the ensemble can find a good approximation to T, which is the point labelled B.

2. Computational Reason: Many learning algorithms, such as neural network and decision tree, are running some forms of local search that only guaranteed to converge to a locally optimal solution. Even with sufficient training data, it may still very difficult computationally for the learning algorithm to find the best hypothesis. Therefore, an ensemble constructed by performing the local search from different starting points may result in a better approximation to the true hypothesis T than any of the individual classifiers as shown in Figure 2.8b.

3. Representational Reason: Given sufficient training data, most machine learning applications such as neural networks and decision trees are very flexible, and they will search the space H for all possible hypotheses (classifiers). However, this exploration process is bounded by a finite training data, and it will terminate when the algorithm finds a hypothesis that fits the training data. The representational issue arises when the true hypothesis T can not be represented by any of the possible hypothesis in H as shown in Figure 2.8c. Therefore, constructing weighted aggregates of hypotheses drawn from H may expand the space of possible functions to find the best hypothesis (B), which is a good approximation to T. Figure 2.8c illustrates this situation.

Generally speaking, a learning algorithm that suffers from the statistical issue can be described as having a high 'variance', a learning algorithm that suffers from the computational issue is often said to have a high 'computational variance', and a learning algorithm that suffers from the representational issue is said to have a high 'bias' [92].

2.6 Ensemble Learning Methods Characteristics

Considering the potential advantages of ensemble systems, it is not surprising that a wide variety of ensemble techniques is available to researchers. According to [73], a typical taxonomy to categorise ensemble methods in classification tasks contains four factors. As Figure 2.9 shows, these factors are: (1) Inter-classifiers relationship, (2) diversity generator, (3) combining method, (4) ensemble selection. Each one of these factors will be discussed in a separate section.

2.6.1 Inter-classifiers Relationship

This ensemble method characteristic indicates whether the base classifiers are dependent or independent. Using this factor, ensemble methods can be categorised into two main types: sequential and parallel.

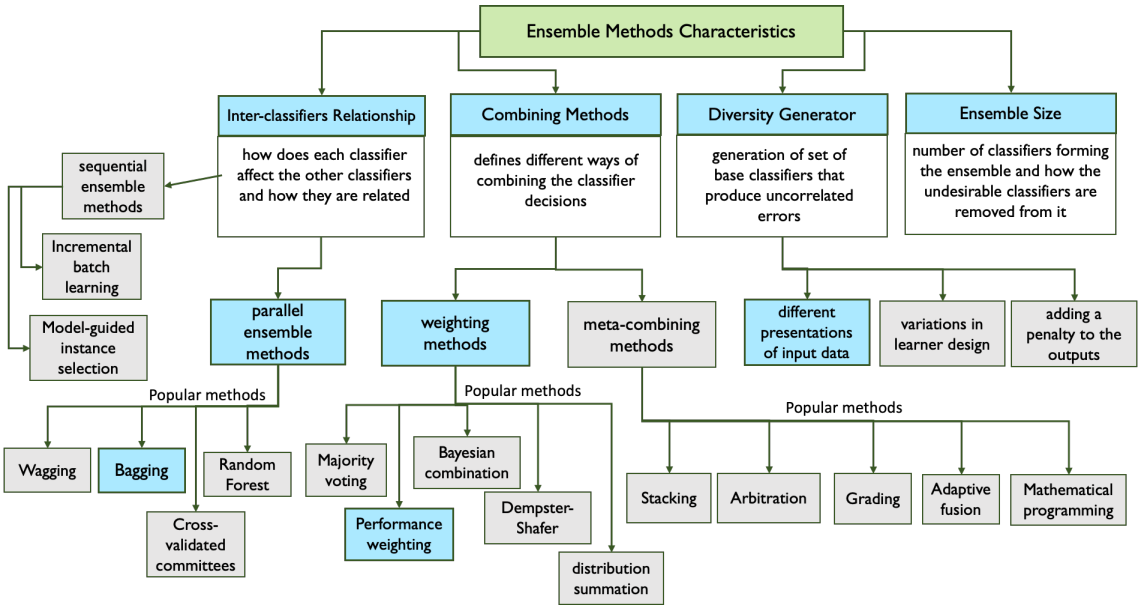
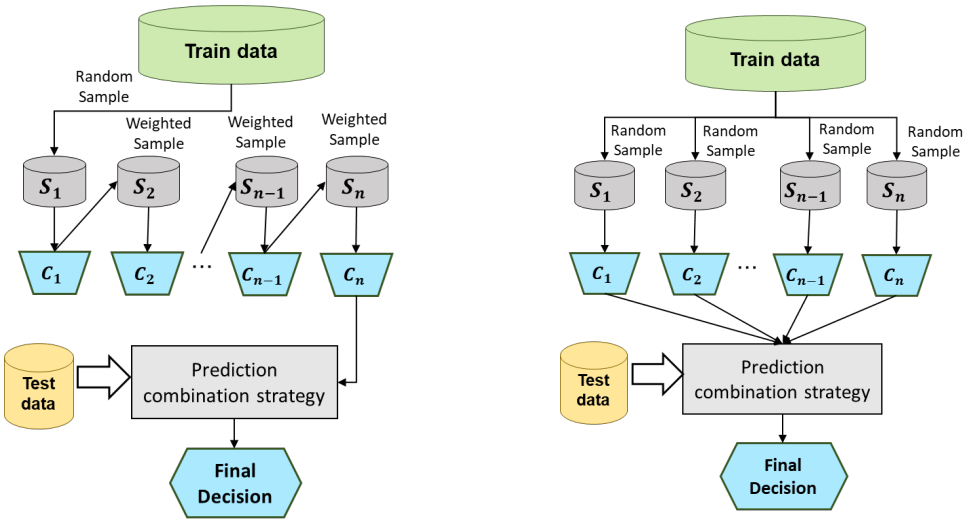


Figure 2.9: Taxonomy for characterizing ensemble methods in classification tasks, blue colour refers to the approaches related to this thesis

2.6.1.1 Sequential Ensemble Learning Algorithms

It is also known as *dependent approaches* where the output of a certain member in the ensemble system influences the training process of the following member. This influence may be recognised by modifications of either the dataset or the algorithm. As shown in Figure 2.10a, there is an interaction between the learning runs. Hence, in each iteration, a continuously refined model may be generated based on the results of the previous executions and also may provide knowledge that can be utilised to enhance the learning process in the next iteration [28]. The idea is to reduce the redundancy and improve the quality of the final results. Therefore, some individual learners will be discarded. Among the popular sequential ensemble methods are boosting, Windowing, and Stacking, which are briefly described below.

Boosting. It is a widely used method, and it works on the principle that a set of weak base classifiers can be boosted sequentially to a strong ensemble classifier. The basic idea of boosting was developed by Schapire in [93] and it described how a gradual combining of multiple classifiers can achieve a higher combined accuracy than those obtained if the base classifiers were used individually. AdaBoost algorithm, which stands for (Adaptive Boosting) is an improvement on boosting and developed in [94]. It starts by assigning an equal weight to each training instance, and then generates the first model on a random sample of the training data. Next, depending on the performance of the first learner, the weights of the training instances of a subsequent learner are adjusted as can be seen in Figure



(a) Sequential System

(b) Parallel System

Figure 2.10: Ensemble Learning Paradigms

2.10a. Specifically, the weights are increased for the misclassified instances and decreased for the correctly classified ones. This process continues until a specified threshold is met or a number of base learners are generated. The idea is to give higher weights to 'difficult' instances, and thus enable the following weak learners to focus more on instances that have been incorrectly classified in previous iterations. Despite that these gradual corrections of the mistakes made by weak models improve the overall classification accuracy of the model, AdaBoost classifier is known to suffer from overfitting especially on noisy datasets [95].

Windowing. It was originally designed to speed up C4.5's tree induction process to deal with large datasets. Windowing starts by selecting a random subset of training dataset, a 'window', in order to build an initial tree. The remaining training instances are used as a validation dataset to test the performance of this first tree. If the accuracy obtained are below a target threshold, all the misclassified instances are removed from the current validation dataset and added to the window in which a new tree is constructed, and then validated using the remaining training instances (non-window dataset). This process is repeated until sufficient accuracy is obtained from the last trained tree, which is considered to be the final classifier. Comparing with other ensemble methods such as boosting, windowing approach has not receive much attention due to the vast development of computer capabilities in terms of memory sizes and processor speeds, which makes constructing a model using multiple rounds of windowed data is not necessarily faster than using a single round with all the data [24, 96]. Also, it has been found in [97] that on noisy domains, windowing approach should be avoided.

Stacking. It was introduced in [98] and unlike boosting, staking is generally used to combine models of the same type, i.e. a heterogeneous method [99]. As the term suggests, it consists of multilevel hierarchy of models. Stacking works by creating a *meta-learner*, which replaces the voting procedure in boosting in order to combine the output of the base classifiers. The process starts by generating what is called *meta-data* based on the complete training data using leave-one-out procedure. Then the meta-learner is trained on the meta-data. The attributes used to build the meta-data, and the algorithm used in the learning procedure, are the most important decisions concerning stacking [10]. The number of levels and the number of models may have also a considerable impact on the final ensemble performance.

2.6.1.2 Parallel Ensemble Learning Algorithms

It is also known as independent approaches, in which each member in the ensemble is generated independently using a different sample of training data. As shown in Figure 2.10b, the base models hardly need to share any information during the induction processes. Instead, collaborations between them are taking place in the testing stage, i.e. to classify a test example each base classifier produces a prediction and all the predictions are combined to decide the final prediction of this example. The idea of parallel ensemble methodology is to improve the classification accuracy of the system by combining different classifiers that commit different errors at different times, according to [100]. Another benefit of the parallel ensemble methods is that they are perfectly suitable to parallel computing, in which the speed and memory constraints can be addressed by distributed environments. Three parallel ensemble methods, namely; Bagging, Random Forest, and Random PRISM are briefly described in the following paragraphs.

Bagging. It is the short name of Bootstrapped aggregating, which is a widely used method developed by Breiman in [9]. In contrast to boosting, bagging works by creating multiple random data samples produced using the strategy of sampling with replacement the original training set as shown in Figure 2.11. These data samples typically have the same number of instances the original training set has. Hence, statistically, each bootstrap sample is likely to have approximately 63.2% of the total number of training instances appearing at least once in the sample [9]. On each sample, an individual base learner is trained by applying the base learning algorithm. Also, approximately 36.8% of the original training instances may not be included at all in the bootstrap sample [9]. They are referred to as the *out-of-bag (OOB)* instances and can be used to validate the base model. Then, to construct an

ensemble learner, bagging typically adopts majority voting to combine all the base classifiers' predictions of a new example. However, other combination methods, such as weighted voting are also possible. It is concluded in literature that bagging techniques are empirically and theoretically can reduce the model overfitting error, and can be more powerful when dealing with unstable models, where small details of the training algorithm/process may lead to significant changes in the model predictions.

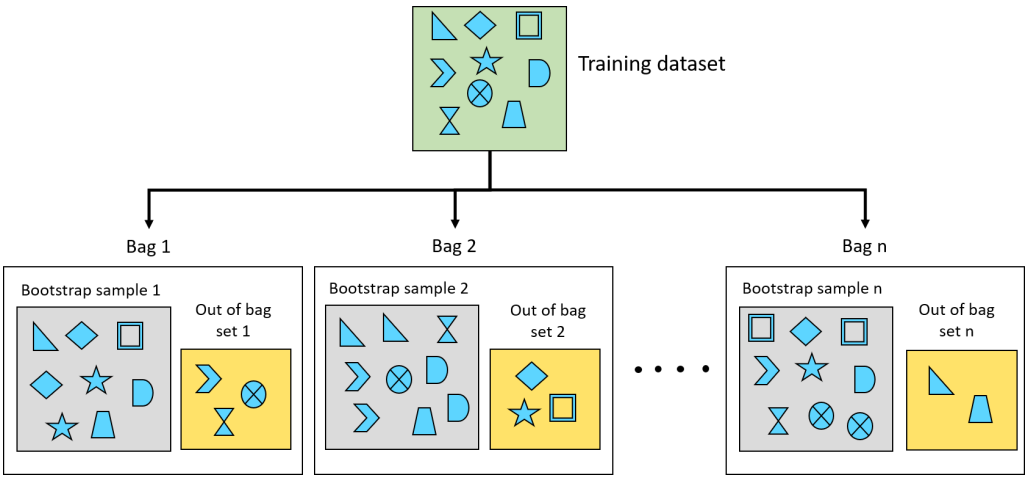


Figure 2.11: Bootstrap sampling with replacement in bagging

The effectiveness of bagging might be explained by its original design as a variance reduction technique, which is mainly obtained through the bootstrap sampling method. It is based on the idea that if the variance of a prediction is σ^2, then the variance of the average of k independent and identically distributed predictions is reduced to $\frac{\sigma^2}{k}$. Given sufficiently independent predictions, such an approach of averaging will reduce the variance significantly. Generally speaking, the bagging method can be used in principle with any kind of data mining algorithm to construct an ensemble classifier. However, the main criticisms of the bagging and other ensemble approaches are the lack of interpretation and the high of computational cost.

Random Forests (RF). It can be considered as a variation of the bagging algorithm constructed from decision trees, i.e. an ensemble of decision trees developed by Breiman in [101]. Briefly, a random forest combines the two concepts of *bagging* and *Random Selection Features*. The latter concept was developed by Ho in [102, 103]. Ho argues that on high dimensional datasets, traditional trees cannot be grown to arbitrary complexity without risking a loss of generalisation caused by high possibilities of overfitting to the training data. Therefore, Ho proposes Random Decision Forest (RDF) approach that generates multiple trees in

randomly selected subspaces of the feature space. Ho's empirical study concludes that these individual trees in different subsets of features generalise their output in integral ways, and hence their combined classification performance is improved.

Just like in bagging approach, a typical RF uses bootstrap sampling with replacement strategy to produce different training sets. Each bootstrapped sample can be used to train an individual tree. Additional randomness can be explicitly inserted into the random forest construction by incorporating the RDF approach in each node splitting decision of each tree. In other words, each time a split in a tree is considered, a random subset of features is selected as split candidates from the overall feature space [104]. Similar to the bagging algorithm, RF uses the out-of-bag (OOB) instances (about 36.8% of the training data) to estimate the test error of each tree. To classify a new unlabelled instance, each decision tree provides a prediction and RF aggregates these predictions and chooses the most voted one as the final classification label for this instance.

Generally speaking, most studies argue that integrating the feature randomness and bagging methods in a random forest model is more likely to ensure building of robust and uncorrelated forest of trees which often leads to better predictive accuracy of unlabelled new instances [105–107]. However, an extensive evaluation study conducted in [107] shows that RF algorithm suffers from several weaknesses. Firstly, RF requires to construct a number of base learners (trees) in the range of 100 to 500 in order to significantly improve the predictive accuracy of the classification output. This might not be a practical solution in the real life applications where retrieving a fast classification decision is critical. Secondly, RF algorithms are likely to build highly-correlated complex trees from high dimensional datasets, which could considerably increase the complexity and the forests' error rate. Thirdly, RF does not consider feature interaction (relationships) that might occur in the feature space.

Random Prism. It is an ensemble learner not based on decision trees but on rule sets produced by PrismTCS algorithm [108, 109]. It follows the parallel ensemble learning approach and uses bootstrap sampling with replacement technique to partition the training data into multiple data samples. Each sample typically has the same size as the original training dataset, and thus, as in bagging methods, some training instances may appear more than once in the same sample and some may not be included at all, i.e. OOB instances. As Figure 2.12 illustrates, each sample trains a different classifier using PrismTCS algorithm and its OOB instances are used as a validation data for this particular base classifier. Once a defined number of base classifiers is induced, Random Prism adopts a weighted majority voting technique to combine their individual predictions of a new unlabelled instance.

It has been empirically demonstrated in [108] that Random Prism outperforms its stand-alone base classifier with regard to accuracy and tolerance to noise. However, also pointed out in [108], Random Prism's computational cost is high. Although this challenge may be common for most ensemble learners. The authors of [110, 111] show that the CPU requirements of Random Prism in terms of time and apace are significantly higher even with modest sized datasets. Therefore, a parallel version of Random Prism has been developed in [111]. It is based on data parallelisation and utilises *Google's MapReduce* programming paradigm [112]. Specifically, Parallel Random Prism distributes the construction of each individual base classifiers to different machines in a computer cluster using the *Hadoop* implementation of MapReduce.

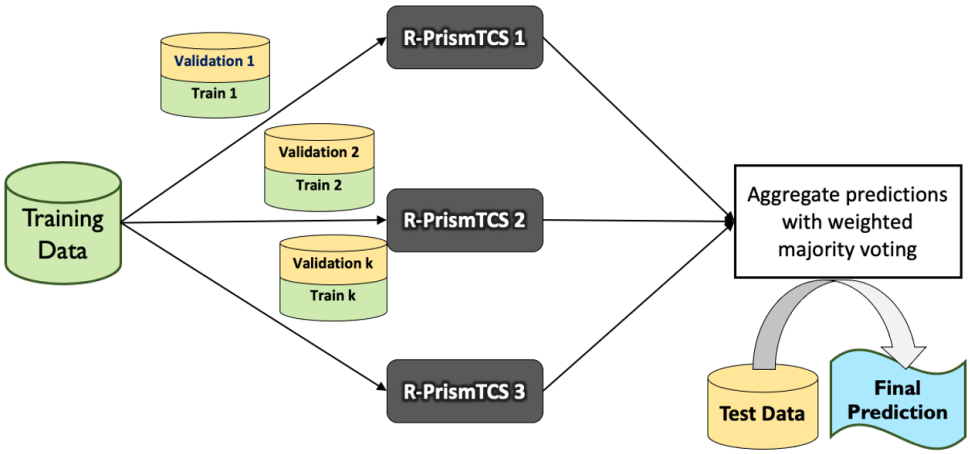


Figure 2.12: The Random PRISM architecture. (adapted from [108])

The authors of [110] provide theoretical complexity analyses of Random Prism algorithm and its parallel version. The experimental study concludes that Parallel Random Prism scales well on a large number of training examples, a large number of data features and a large number of processors. Furthermore, Random Prism is an *accuracy-oriented* ensemble model, which uses each base classifier's accuracy to assess the quality of its participation in classifying a new unseen instance. However, this aspect has been criticised by several studies such as [113], which found that the accuracy is unreliable as a measure of a classifier's quality especially for unbalanced datasets.

2.6.2 Diversity Generators

As previously illustrated in Figure 2.9, diversity generator is one of the ensemble method characteristic, which refers to the generation of a set of base classifiers that should be as diverse as possible to assure producing uncorrelated errors, and then obtain more accurate ensemble [24]. It has attracted a lot of attention in

the literature, because of its significant impact on the performance of ensembles. There is no explanatory theory that defines why and how diversity among individual classifiers contributes to overall ensemble accuracy [24, 114, 115]. An attempt to investigate the hypothetical relationship between diversity and the ensemble accuracy have been found in [25]. This study uses an extensive example of an ensemble of three classifiers to try to predict 10 new instances. Each classifier predicts exactly 6 of the 10 objects (individual accuracy $= 0.6$) and thus 28 possible distributions of correct/incorrect votes are produced. These two limit lists were termed as 'pattern of success' and 'pattern of failures' respectively. The experiment resulted to that the accuracy of the ensemble varied between 0.4 and 0.9. The reader is referred to [25] for further details about this experimental study.

Moreover, surveys; such as that conducted in [115] reviewed the various techniques used for creating diverse ensembles and categorise them according to whether they explicitly employ diversity using some measurements, or whether they implicitly inject randomisation methods into the system to encourage diversity. Several studies such as [92, 115] describe the popular methods used for creating diversity in ensembles as follows:

1. Manipulating the Inducer - this method attempts to diversify individual learners by using different parameter settings for the base learning algorithm, i.e. each ensemble member is trained by an inducer that is manipulated differently

2. Manipulating the Training Samples - in this widely used method, each classifier is trained on a different subset of the original dataset. This category of diversity creation includes bagging and boosting algorithm. However, the data sample manipulation depends on the sampling approach itself, e.g. bagging employs bootstrap sampling with replacement, and boosting utilises sequential sampling.

3. Manipulating Input Feature - the training data is usually described by a set of features. Different samples of features give different representations of datasets. Therefore, in this method, each base learner trained from different subsets of features. However, this method may not appropriate for datasets with only a few features.

4. Manipulating Output Representation - in this method that manipulates the target attributes, instead of generating a single complex learner, an ensemble of multiple learners with different and often simpler representations of the target attribute can be generated. It is based on a concept called 'aggregation' [24].

It is important to note that the above popular diversity generation methods can be used together, e.g. Random Forest [101] employs both methods of training samples manipulation and input feature manipulation.

2.6.3 Combining Methods

As previously illustrated in Figure 2.9, combining methods are one of the ensemble method characteristic. 'Fusion Methods' is the alternative term of the combining methods in a number of studies. Simply, for classification tasks instead of trying to determine the perfect single model, combining the outputs of a diverse set of models can mostly achieve accurate prediction ability. This is the main philosophy of ensemble systems and the fundamental theoretical reasons behind that explained in Section 2.5.1. The combining strategies can be divided into two main groups: (1) weighing methods, (2) meta-combining methods. Only the first group will be briefly described in the following paragraphs, as they are within the scope of this thesis. However, detailed explanations about the second group of combination methods (meta-combiners), can be found in [24].

As their name implies, weighing methods assign weights to each classifier's prediction, and then use these weights to produce the final ensemble classification. The combination methods belong to this group include: majority voting, performance weighing, Bayesian combination, Dempster–Shafer, and distribution summation, etc.

Majority Voting. In this widely used type of voting, all the base classifiers are treated equally, and thus equal weights are assigned to each classifier's output. The ensemble learner performs the combined classification for a new unlabelled instance according to the class that obtains the most frequent vote. Several ensemble classifiers adopt this equal voting, such as Random Forest [101, 102].

Performance Weighing. The weighted majority voting is the alternative name of this type of combination in the literature. It is among the most widely used combiners [25]. In contrast to the majority voting, weighted majority voting considers the performance evaluation of each base classifier on a validation dataset as a weight to avoid a potential problem of reliability that may occur in simple majority voting method, when some base classifiers are more reliable than others. Hence, each base classifier's decision is multiplied by its weight to reflect the confidence of the decision. Giving the qualified models more power in making the final classifications of an ensemble would improve the overall predictive performance of the ensemble [24].

Bayesian Combination. In the Bayesian combination method, each base classifier has been assigned its posterior probability as a weight given the training set. Then, to make a prediction of a new unlabelled instance, the ensemble choose the classifier's decision with maximum posterior probability. The full learning process is outlined in [116]. Also, the interested reader is referred to [25] for further discussion about this combination strategy.

Dempster-Shafer (DS). This combination method is based on Dempster-Shafer theory [117], also referred to as the *theory of evidence* or the *theory of belief functions*. DS is developed by Arthur Dempster in 1968, and then improved by Glenn Shafer in [118]. The idea of utilising DS theory for combining multiple base classifiers was introduced in [119] by selecting the class that maximizes the value of the belief function. Several combination methods have been inspired by this theory, e.g. a recent approach presented in [120] to solve the motor imagery task classification for Brain-computer interface (BCI) systems, which can be utilised to help disabled people to interact with the world through their brain signals.

Distribution Summation. This combination method is based on the conditional probability vectors, which obtained from each base classifier in an ensemble [62]. Then, to classify a new instance's class, the ensemble selects the class with the highest value in the total vector. The mathematical formula of this distribution is illustrated in [24].

2.6.4 Ensemble Selection

An important characteristic of ensemble methods is to define how many base classifiers should be generated (ensemble size) and which classifiers should be included in the final ensemble model (models selection) [24]. In this subsection, two types of ensemble selection methods will be discussed.

2.6.4.1 Pre-Selection of the Ensemble Size

Several ensemble learning algorithms, such as bagging and Random Forest treated the ensemble size as a hyper-parameter, which is tuned using a quantitative approach. Determining a proper size of an ensemble requires balancing accuracy and efficiency. The size can not be too small because adequate data must be available for each learning process to maintain diversity and generate an affective classifier. Also, the ensemble size can not be too large because it would be very expensive in terms of computations and memory resources [24, 121]. In other words, the number of base classifiers that should be used in an ensemble is usually defined ac-

cording to the following aspects: (1) desired accuracy, (2) computational cost, (3) number of processors available (memory resources), (4) the nature of the classification problem. However, the impact of these aspects on the ensemble efficiency and their strong relationship to each other make the ensemble size determination difficult [114]. Extensive experiments carried out in [121] using two widely used parallel ensembles (bagging and random forest) show that the ideal ensemble size can be very sensitive to the nature of the classification problem. Another major experimental study conducted in [122] suggested constructing between 64 and 128 base learners to ensure a balance between computational cost and accuracy. The same study has shown that there is no significant performance gain if a larger number of base models is used.

2.6.4.2 Post-Selection of the Ensemble Size (Models Selection)

As stated previously in this section, the number of component classifiers that should be included in the final ensemble is an influential factor for building an efficient and accurate ensemble [24,114]. A large ensemble explores different feature subspaces, which might increase its general classification accuracy. However, it requires a higher computational overhead than a smaller one, and it decreases the ensemble's explainability. To overcome this trade-off, reducing the ensemble size should be considered, but to what extent this reduction can be applied without causing significant accuracy loss to the whole model is difficult to determine.

According to an empirical study presented in [123], a compact ensemble can be extracted from a large one without reducing the whole ensemble's predictive performance in terms of diversity and accuracy. Moreover, the theorem of *'many could be better than all'*, which was presented in [124] inspired researchers to introduce many ensemble selection methods. Roughly speaking, the two most popular approaches for selecting an ensemble subset are: (1) Ranking-based method, and (2) Search-based method. Both approaches will be presented in the following paragraphs, however the former will be illustrated further because it is related to the models' selection method that have been developed in this research. The reader is referred to [125] for greater detail and also additional models selection approaches.

Regarding (1), the main concept of *Ranking-based approach* is to separately rank the base classifiers according to a certain criterion, and then choose the top classifiers that have ranks above a certain threshold [24]. The traditional ranking approach is based on the individual model's accuracy on a separate validation dataset. However, this is not an adequate measure, especially in case of dealing with imbalanced datasets [113,126]. A considerable amount of literature has been

published on investigating various ensemble selection methods; however, there is no particular way to identify which classifier will produce the best prediction on a particular dataset.

For example, a ranking-based method proposed in [127] suggests ranking classifiers according to their classification performance on a different validation dataset and their performance to correctly classify certain classes. Another study proposes generating a selective ensemble of rough subspaces by utilising an accuracy-guided forward search strategy to add the most relevant members gradually into the ensemble system. The algorithm is termed, FS-PP-EROS and the reader is referred to [128] for more explanation. The third example of ranking-based ensemble methods is also a diversity measure called Kappa pruning, which was introduced in [129]. In this method, the Kappa statistic k is computed for all pairs of classifiers. Pairs of classifiers are chosen in ascending order of their concurrence level until a desired ensemble size is obtained. Last example of ranking-based techniques is orientation ordering [130] for base classifiers obtained from bagging and then selecting a subset for combination. The selected sub-ensemble containing between 15% and 30% of the original ensemble size. Moreover, an important issue arises in most ranking-based ensembles following employing a ranking process is the final number of base classifiers to choose. The common ways to obtain this fixed number are a user-defined input or percentage of the components.

With regard to (2), instead of separately ranking an ensemble's members, *Search-based methods* perform a heuristic search in the space of the possible different ensemble subsets while evaluating the collective merit of a candidate subset" [35]. Among others, GASEN algorithm [124] is a neural network ensemble learning system that follows this method. The algorithm starts by training multiple neural networks, and then assign a random weight to each of these neural networks. Next, GASEN utilises a genetic algorithm to develop and update those weights so that they can define the quality of each individual neural network. Finally, it selects the networks with weights bigger than a pre-defined threshold to make up the final ensemble model. The authors of [124] argue that GASEN algorithm can not only construct neural network ensembles with far smaller sizes than those created using bagging or boosting algorithms, but also achieves stronger generalisation ability. Another study introduced in [131] proposes ranking the base classifiers according to their ROC performance, and then evaluate the ensemble model using the top-ranked members. Then, additional members are gradually added to the ensemble subset since its performance are continuously improving. Hence, the ensemble size will be sequentially increased until its performance does not improve further.

2.7 Evaluating Ensemble of Classifiers

Evaluating the performance of an ensemble is very important for assessing the quality of the ensemble and regulating its parameters accordingly. There are several criteria for evaluating the predictive performance of ensembles systems, such as computational complexities, interpretability of the resulting ensemble, scalability to large datasets, and robustness, etc. This section reviews some of those performance measures.

2.7.1 Computational Complexity

Generally speaking, computational complexity of machine learning models includes 'time complexity' and 'space complexity'. The former refers to the amount of CPU time consumed by each classifier, and the latter represents the amount of extra memory required to implement the classifier. According to [24], there are three common metrics that can be used for computational complexity evaluation:

1. Computational complexity for generating a new classifier, especially when dealing with massive datasets.

2. Computational complexity requires to update the current classifiers given new data added.

3. Computational complexity that is needed to classify a new unseen instance.

Despite the importance of creating an ensemble with the highest possible accuracy, reducing the computational complexities is necessary to increase the ensemble efficiency. Moreover, the smaller or compact models demand less space (memory), which is particularly crucial in multiple real-time applications.

2.7.2 Interpretability of the Resulting Ensemble

It is also known as the comprehensibility criteria of the ensemble model. Interpretability is the degree to which the final ensemble results can be understood by humans. This is particularly essential in critical applications, such as medical diagnosis [132] where a wrong decision can have very serious consequences. Interpretability is treated in the literature as a subjective issue that is not clear how to be measured. However, there is a number of indicators and quantitative measures that can help in evaluating these criteria. For example, *compactness* which can be measured by the ensemble size, and the *interpretability of the base inducer used* can impact the interpretability of the ensemble. Therefore, choosing a base learning algorithm that generates base learners with a good expressive power,

such as rule-based classification algorithms may help to reduce the risk of creating complex ensembles. However, according to [92] this would not be always guaranteed as the comprehensibility lost in the ensemble is also influenced by its adopted combination method.

2.7.3 Scalability to Large Datasets

Scalability indicates the ability of the classification algorithm to effectively deal with large datasets, especially that the amount of data in real-life is considerably increasing and this may introduce time and memory problems in most ensemble methods [24]. However, there are ensemble approaches that are more appropriate for scaling to the large datasets than other approaches. For example, independent (parallel) approaches are considered more scalable than dependent (sequential) approaches because of their ability to be trained in parallel using different processors. Moreover, the partitioning methods are more suitable for scaling to large datasets.

2.7.4 Robustness

It refers to the efficiency of the ensemble to process data with missing values or to deal with noise. The robustness of an ensemble is commonly measured using the following process: (1) train the ensemble on a clean dataset, (2) add some artificial noisy instances to this clean dataset to produce a noisy training set, (3) train another version of the ensemble using this noisy dataset, (4) compute the accuracy of the two ensemble learners. The robustness level is measured as the differences in the performance of those learning processes.

2.8 Summary

The goal of this work is to develop an efficient, accurate predictive ensemble learner that exhibit a similar expressiveness as its single base learner. Therefore, the literature concerning the two paradigms: (1) single predictive learning, and (2) ensemble predictive learning, has been thoroughly reviewed in this chapter. Several existing approaches in both paradigms have been assessed for their suitability to be employed in constructing the ensemble system that can satisfy the research question and objectives.

In terms of (1), Section 2.2 has examined a number of widely used predictive data mining algorithms such as decision tree, rule induction, support vector machine, k-nearest Neighbors, and artificial neural network. Approaches to classification rule induction have been surveyed further in Section 2.3, e.g. AQ family of algorithms, CN2 algorithm, RIPPER algorithm, and PRISM family of algorithms, etc.

Regarding (2), predictive ensemble learning methods have been illustrated in Section 2.5 and Section 2.6.1. Approaches to generating an ensemble model can be categorised into two main types: sequential and parallel. Section 2.6.1.1 has explored a number of sequential approaches such as Boosting, windowing, and stacking, while the parallel approaches such as Bagging, Random Forest, and Random Prism have been discussed in Section 2.6.1.2.

The performance of an ensemble model is highly depended on the level of diversity among the group of classifiers that constitute the ensemble. The widely used diversity generation methods are explained in Section 2.6.2. The combination strategy that can be utilised by an ensemble model is also an essential factor to produce not only accurate, but more robust classification results. Several popular ensemble combining methods; such as majority voting, weighted majority voting, Bayesian combination, Dempster-Shafer, and distribution summation are introduced in Section 2.6.3.

Moreover, defining (1) how many base learners should be generated, and (2) which members should be included in the final ensemble model is another important characteristics of ensemble methods. Regarding (1), which is also called a 'pre-selection of the ensemble size' and it can be tuned as a hyper-parameter using a quantitative approach, has been highlighted in Section 2.6.4.1. Concerning (2), which is also known as 'models selection' has been reviewed in Section 2.6.4.2.

Furthermore, evaluating the performance of an ensemble model is also crucial for assessing the quality of the ensemble and regulating its parameters accordingly. A number of criteria for ensembles evaluation such as computational complexity, interpretability, scalability to large datasets, and robustness are discussed in Section 2.7.

Specifically, the literature analysis carried out in this chapter leads to the following concluding remarks:

- Modular rule induction, PRISM family of algorithms in particular have shown to be superior with regard to the expressiveness level comparing with most classification algorithms. Further investigations on this will be provided in Chapter 3.

- Bagging is a robust ensemble method, and it is generally designed as a variance reduction technique to reduce the model overfitting error, which can be

effective when dealing with unstable weak classifiers. Bagging as a parallel (independent) ensemble approach is, also more suitable for scaling to the larger datasets than dependent (sequential) ensemble approaches, in which it can address the speed and memory constraints by distributed environments. More discussion on the benefits of using bagging in decision-making applications will be provided in Chapter 5.

- Combining multiple predictions from diverse classifiers using weighted majority voting technique are more reliable than other techniques, because it gives the qualified models more power in making the final classification, which would improve the overall predictive accuracy of the ensemble model.

Chapter 3

New Modular Rule Induction Approaches for Continuous Attributes

This chapter discusses and analyses a number of limitations existing in separate and conquer approaches, particularly in PRISM family of algorithms. The computational issues with converting continuous attributes into categorical ones using discretisaton methods are investigated. Then, the chapter introduces a more efficient and expressive method to induce numeric rule-terms from continuous attributes. The method is used to develop two new members in the PRISM family of algorithms (G-Prism-FB and G-Prism-DB). These algorithms are among the contributions of this project and are published in [19] and [20], respectively. They are also, empirically evaluated in this chapter.

3.1 Introduction

In the previous chapter, a number of predictive data mining approaches were identified from the literature that could be used to meet the project's aim, which is to develop an efficient and accurate predictive ensemble learner that exhibit a similar expressiveness as its single base learner while improving its accuracy. These approaches were examined for their suitability as a base learning algorithm (inducer) for an explainable ensemble system. A choice was made to develop a rule induction approach in relation to the PRISM family of algorithms to be used as an inducer for the ensemble.

The new rule induction classifier must be able to induce highly expressive rules from datasets in order to be in line with the research requirements, objective 2 in particular:

> *"To measure and compare the expressiveness of rule based models and develop an appropriate rule-based predictive algorithm suitable as base learner for an ensemble."*

In Section 3.2, this chapter will begin with the rationale behind the choice of PRISM algorithm derived from Chapter 2. A number of shortcomings found in the current implemented PRISM based algorithms are discussed in Section 3.3. However, the chapter focuses mainly on the problem of dealing with continuous features, which is thoroughly investigated in Section 3.5.1. The common discretisation techniques utilised in rule based algorithms to deal with numerical features are reviewed in Section 3.4.

Then, a more efficient and expressive method to induce rule-terms from continuous attributes is proposed in Section 3.5.2 This new rule-term structure is integrated in the PRISM family of algorithms, for which two new members are developed in Section 3.6. Section 3.7 demonstrates the empirical evaluation of these new algorithms comparing with the original PRISM algorithm. Finally, a short summary of this chapter is provided.

3.2 Modular Rule Induction using PRISM family of Algorithms

The PRISM family of algorithms follows the 'separate and conquer' approach, outlined in Section 2.3.2, to induce expressive modular classification rules directly from training data. In contrast, the 'divide and conquer' rule induction approach, outlined in Section 2.3.1, generates rules from an intermediate form of a decision tree. A rule-based learner is considered more expressive when it produces fewer numbers of rules with less complex terms per rule. This is because a human user can more easily interpret the rationale behind the predictions of a smaller and less complex rule set.

'Modular Rules' refers to the representation of rules that often can not be fit into a decision tree without creating at least one irrelevant feature [5]. This issue that has been previously demonstrated in Figure 2.6 is called 'replicated sub-tree problem', which was recognised by Cendrowska [43] and considered the main cause of overfitting. In fact, she strongly criticised this way of constructing classification rules and argued that tree-based induction algorithms grow needlessly complex,

and this would result in redundant rule-terms, which is completely undesirable in expert systems. Generally speaking, complexity in models often leads to the lack of interpretability issues, which can be very expensive in real world datasets, despite how accurate the model is. This can be explained by the example, previously presented in Section 2.4, which was about an optician who has to decide whether a patient should be fitted with hard or soft contact lenses or there is no contact lenses suitable to the patient. The decision of 'no lenses are suitable' can be taken within a couple of seconds using the single compact Rule 2.3 induced by PRISM algorithm. Contrarily, using a decision tree shown in Figure 2.7 would require the optician to advise the patient to do an unnecessary test of the tear production rate, which would cost time and probably an additional fee to be paid. In fact, the consequences could be even more serious if this avoidable procedure involves surgery.

This level of expressiveness cost was one of the main reasons to choose the PRISM algorithm in this thesis to be the basic rule induction strategy to develop the proposed new modular rule-based algorithms. Another reason for choosing PRISM, is the results of a series of experiments conducted in [5, 69], which were in line with Cendrowska's original findings as they strongly indicate that PRISM-based algorithms not only produce a more expressive compact set of rules than tree-based rules, but also it achieves a similar or better predictive accuracy. On noisy data, the same studies found that PRISM was more stable than tree-based algorithms, even when a high level of noise exists in the training dataset.

The more important reason for choosing PRISM, is related to its ability to abstain from classifying a new example when it is uncertain about the classification and does not take a risk of classifying the instance wrongly, whereas tree-based approaches should always produce a classification because of their hierarchical structure. Abstaining aspect in PRISM family of algorithms may contributes to increased explainablility in the model by ensuring the trustworthiness of the induced rule set. This aspect is necessary in critical applications; such as in medical diagnosis, financial analysis, terrorism detection, etc.

Nevertheless, all members of PRISM family algorithms such as N-Prism, PrismTC, and PrismTCS (see Section 2.4.4), suffer from some practical and computational issues, which will be discussed in the next section. Among these limitations is the high computational cost of dealing with continuous attributes, which is addressed in this thesis and an alternative solution is introduced accordingly in Section 3.5.

3.3 Limitations in the PRISM family of Algorithms

3.3.1 Dealing with Tie-Break and clashes

'Tie-break' occurs when two attribute-value pairs (terms) have equal probability of occurrence. Where the original PRISM algorithm decides to choose arbitrarily, its predecessor (N-Prism [69]) takes the rule-term with the highest total frequency, i.e., covers the highest count of the target class. *'Clashes in training data'* occur whenever there are instances in a subset of the training set that are assigned to different classes, but sharing the same attribute values, and hence cannot be separated further. Such a subset is known as a 'clash set'. The implemented version of PRISM [70, 71] addresses the problem of clashes as follows:

- Determine the clash set's majority class.
- Check if the majority class is also the target class of the rule currently being generated. If this is the case, the rule is completed for the classification of the target class. If it is not, the rule is discarded.

The description in [69,70] does not provide any additional instructions about how to deal with the clash set. Note that not handling the clash set properly means that PRISM would be trapped in an infinite loop as the same rule would be generated, from the same clash set, all over again and again discarded [133]. Therefore, Bramer in [133] explained a strategy to handle the clash set. The strategy is to remove all the instances that belong to the discarded rule from the clash set. This saves Prism-based algorithms from encountering the same clash set all over again.

3.3.2 Dealing with Conflict Resolution

'Conflict resolution' issue occurs in most separate and conquer rule induction algorithms. It is also known as classification conflict, i.e. an unseen instance covered by several rules with different classification outputs (predictions) in the same rule set. According to [5], the strategies that can be used to solve this conflict include:

- 'Majority voting', e.g. an instance fired three rules predicting class d and one rule predicted class a, so the final classification will be d.
- 'Giving priority' to certain types of rules or classifications, e.g. rules with fewer number of terms or rules predicting a minority class, etc.
- 'Take the first one that fires'. This is the widely used strategy, even though it is very basic. The main advantage of this method is the considerable reduction in the amount of processing required. However, it makes the order of the rules in a set very important. The current research adopt this strategy.

3.3.3 Dealing with Missing Values

In many real-world datasets, not all attribute values are recorded, and thus learning algorithms have to be able to deal with missing information and unknown values. There are several strategies for handling missing values and according to [3, 5], the most commonly used are:

- *'Delete strategy'*, which is completely ignoring all instances that have at least one missing value. Applying this method is very simple, and it does not necessitate any changes in the learning algorithm. Also, it has the benefit of avoiding introducing any noise and data errors; however, it is wasting training instances and impacts the reliability of a model's results negatively, especially if it applied to a large proportion of the data. Therefore, the algorithms developed in this project do not support this strategy.

- *'Common value strategy'*, which is based on estimating each of these missing values using the most frequently occurring (non-missing) value for categorical attributes, and the average value (the mean) for continuous attributes. The former way seems to be reasonable if the values of the target attribute are very unbalanced but if they are evenly distributed, the reasonableness of this method would be doubtful. In terms of continuous attributes, it is a straightforward and effective way [5] however, like any other strategy for handling missing values, it has to be utilised with care to avoid incorrect results, especially when the number of unknown attribute values is high.

3.3.4 Dealing with Continuous Attributes

Generally speaking, many data mining algorithms, including TDIDT tree based, Naive Bayes, CN2, C4.5, PRISM, etc. require all the attributes to be categorical. However, most real world applications of learning involve attributes that are naturally continuous, e.g. length, weight, temperature, speed, etc. A common way to handle this issue is to convert continuous features into categorical ones, i.e. known as *'discretisation'*. Therefore, several methods have been developed in the literature for this purpose. The next section thoroughly examines a number of these methods including the method utilised in PRISM family of algorithms [69, 71], which is based on a local discretisation approach called 'cut-point calculations' or 'binary discretisation' [134]. Although, this discretising technique was allowed PRISM based algorithms to deal with continuous attributes, major concerns about its computational efficiency have arisen [5]. This will be more thoroughly investigated in Section 3.5 in which a new alternative numerical rule-term structure based on probability density distribution is proposed.

3.4 Discretisation Techniques

Discretisation is the process that transforms the continuous or numerical attributes into categorical ones by partitioning the range of numeric variables into sub-ranges (intervals). Each interval is labelled a discrete value, i.e. a categorical value and then the original numeric data will be mapped to a set of discrete values [135]. It is required in a large number of machine learning and statistical techniques that can only be applied to categorical data. The goal of discretisation is to find a set of cut-point that can create a number of intervals that have a good class uniformity [136]. A typical discretisation process generally consists of four steps: (1) sorting (in an ascending or descending order) the numeric values for the target attribute to be discretised, (2) selecting the best cut-point to split or to merge two adjacent intervals, (3) discretising the numeric attribute values continuously by splitting or merging intervals depending on the type of discretisation, (4) specifying a stopping criterion to end the discretisation process [137].

3.4.1 Categorization of Discretisation Approaches

Generally speaking, the discretisation algorithms can be categorised from multiple perspectives, which include, (1) local and global, (2) top-down and bottom-up. These categories can be (briefly) described as follows:

Local and Global Discretisation Approaches:

The methods belong to this category are distinguished by the timing of the discretisation process. Local discretisation methods, such as 'cut-point calculations' or 'binary discretisation' [134] carry out discretisation during induction; whereas global methods, such as 'ChiMerge algorithm' [138] and 'CAIM algorithm' [139] convert each continuous attribute to a categorical once and for all as a pre-processing step prior to the induction stage. Both approaches need a stopping criterion to terminate the process at some point. Notice that all the aforementioned discretisation algorithms have been implemented in this project for comparative purposes and therefore, they will be further explored in the next section[1].

[1]All the source codes are available in a public online repository at `https://github.com/ManalAlmutairi/PhD_Project_Codes/tree/v1.0.0` and are archived at `https://doi.org/10.5281/zenodo.5557590` [23].

Top-down and Bottom-up Discretisation Approaches:

In another perspective, the top-down and bottom-up are characterised by behaviour of the discretisation process. The top-down starts with one big interval consisting all known numeric values and then continuously splitting this interval into multiple ones. Cut-point calculations and CAIM approaches follow this discretising strategy. In contrast, bottom-up technique such as ChiMerge algorithm begins with a number of intervals and then evaluate their cut-points; these individual intervals are continuously merged until a stopping criterion is met.

3.4.2 Discretisation Algorithms

Cut-point Calculations (Binary Discretisation):

It is a local top-down discretisation technique, and thus each continuous attribute is converted into a categorical attribute at each stage of the induction process. To the best of the author's knowledge, there is no particular source cited in the literature as the main developer of this discretisation method although it has been used, for years, in several data mining approaches such as in [27, 30, 47, 55, 134, 140]. However, as mentioned in Section 3.3.4, this method of handling continuous attributes was integrated into PRISM family of algorithms by Bramer in [69, 71]. Detailed investigation of this will be provided in Section 3.5.1 with examples.

In general, binary discretisation is simply a logical condition, in terms of one or more attributes that can partition the data into at least two subranges [134]. Assume that a threshold value, v, for a continuous attribute α is determined by computing the information gain (or other measure) of multiple values in α (cut-points) and choosing the one with the largest information gain. Accordingly, the threshold value v splits the training data into two parts joined by the logical 'and' operator, those instances for which $\alpha \leq v$ and those for $\alpha > v$.

ChiMerge Algorithm:

ChiMerge is a global discretisation technique developed by [138]. It consists of initialisation step and a bottom-up merging process, which is continually running until a particular threshold is exceeded. ChiMerge utilises a statistical technique, namely the chi-square (X^2) value, which can be computed using the formula shown in Equation (3.1).

$$X^2 = \sum_{i=1}^{2} \sum_{j=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \tag{3.1}$$

Where:
$k =$ number of classes,
$A_{ij} =$ number of examples in the i^{th} interval, j^{th} class,
$E_{ij} =$ expected frequency of $A_{ij} = \frac{R_i * C_j}{N}$,
$R_i =$ number of examples in i^{th} interval = $\sum_{j=1}^{k} A_{ij}$,
$C_j =$ number of examples in j^{th} class = $\sum_{i=1}^{m} A_{ij}$,
$N =$ total number of examples = $\sum_{j=1}^{k} C_j$

Each continuous attribute in a dataset is individually discretised only once using the following steps:

1. Sort the values of a target continuous attribute into ascending order.
2. Calculate the X^2 value for each pair of adjacent intervals as shown in equation (3.1).
3. Merge the pair of adjacent values that has the lowest X^2 value.
4. Repeat steps 2 and 3 until a user-defined (X^2 threshold) is exceeded.

According to [138], ChiMerge is a robust discretisation algorithm and the main source of robustness is that taking the class of an instance into consideration when generating or adjusting intervals. Also, ChiMerge works well in practice and can be applied to multi-class learning [5]. However, the algorithm has several problems, e.g. it tends to construct too many intervals; and since it is a parametric algorithm, setting a high (X^2 threshold) value may eliminates important intervals. A serious weakness is its lack of global evaluation, as each attribute is discretised individually without considering the other attributes. Moreover, the merging process in ChiMerge can only examine the two adjacent intervals and ignoring other intervals. The interested reader is referred to [5, 138] for more details.

CAIM Algorithm:

It is a global top-down discretisation technique developed by [139] and based on the interdependency between class and attribute values to determine the width of every interval. The main goal of CAIM approach is to increase the attribute-class relationship and therefore decrease the number of discrete intervals generated. The dependency between the class C and the discretisation variable D for attribute F can be measured using the heuristic criterion shown in Equation (3.2).

$$\textit{CAIM}\,(C, D|F) = \frac{\sum_{r=1}^{n} \frac{max_r^2}{M_{+r}}}{n} \tag{3.2}$$

Where:
C is the class variable, D is the discretisation variable,
F is the attribute, n is the number of intervals,
r iterates through all intervals, i.e. $r = 1, 2, ..., n$
max_r is the maximum value among within the rth column of the quanta matrix,
M_{+r} the total number of continuous values of attribute F.

Like other global discretisation methods, CAIM starts discretising process by selecting a continuous attribute and sort its values ascendantly. Generate a single large interval contains all the possible values of the continuous attribute and then divides it repeatedly. The algorithm computes the CAIM criterion of all the possible division boundaries produced and selects the highest value. The main advantage of CAIM algorithm is that it does not require the user to provide any parameter. However, the algorithm is computationally very expensive as it requires testing of all possible cut-points. Also, CAIM gives a high factor to the number of generated intervals when it discretises an attribute, and thus the number of intervals would be almost close to the number of classes. Moreover, for each generated interval, only the class with majority number of instances would be highly considered, while the other classes would be mostly ignored. Such a consideration may decrease the quality of the discretisation in some cases [141, 142].

3.5 A new efficient Rule-Term Induction Method for Continuous Attributes

The problem with any method of discretising continuous attributes is that of oversensitivity. Therefore, regardless of the heuristic used to select a particular cut-point, there will always be values that fall just above the cut-point and thus treated very differently from those that fall just below the cut-point [5]. This issue can be considered a general disadvantage of any discretisation method, including the binary splitting approach that has been implemented in PRISM family of algorithms. More investigation on this method is presented in the next section.

3.5.1 Computational Issues with Discretising Continuous Attributes in PRISM Family of Algorithms

As previously stated in Section 3.3.4 and shown in Algorithm 2, the basic PRISM algorithm can only train from categorical data, i.e. only produces rule-terms of the form $(\alpha = v)$ where v is one of the possible values of attribute α. However, in order to overcome this limitation, the other variations of PRISM use a local discretisation method (cut-point calculations) to deal with continuous attributes. Hence, they produce rule-terms of the form $(\alpha < v$ and $\alpha \geq v)$ or the form $(\alpha \leq v$ and $\alpha > v)$. In this case, v is a constant that represents a subrange of numeric values α_j for a continuous attribute α. Generally, the process of how the improved versions of PRISM incorporate this binary splitting approach in their implementations can be described as follows:

- For each continuous attribute α in the training data:
  1. Sort the data into ascending numerical order;
  2. For each possible value in α (v_1 , v_2, ... v_n) where n represents the number of the distinct values in attribute α
     (a) Calculate the conditional probability for a given target class C_i for both rule-terms $\mathbb{P}(C_i|\alpha \leq v)$ and $\mathbb{P}(C_i|\alpha > v)$
  3. Select the rule-term (α_j) with the maximum conditional probability that have been calculated in step 2a;
  4. Add α_j to the left-hand side of the target induced rule R.

Interestingly, the previous described method requires many cut-point calculations for the conditional probabilities for each possible value, v of a numeric attribute α. This would lead to a very high computational complexity in real world applications, which often involve discretising very large datasets.

Table 3.1: Example of a very simple small dataset

Continuous Attribute X	Class Label
4	B
9	A
13	A
22	A
33	B
40	B

For example, Table 3.1 shows a simple small dataset that compromises of just six instances, one continuous attribute and two class labels. In order to induce a single rule-term for class A only, 12 cut-point calculations are required, as shown in the list below (3.3).

$$\left.\begin{array}{lcllll} \mathbb{P}(\text{A}\,|X \leq 4) & = & 0.00 & \textbf{AND} & \mathbb{P}(\text{A}\,|X > 4) & = & 0.60 \\ \mathbb{P}(\text{A}\,|X \leq 9) & = & 0.50 & \textbf{AND} & \mathbb{P}(\text{A}\,|X > 9) & = & 0.50 \\ \mathbb{P}(\text{A}\,|X \leq 13) & = & 0.67 & \textbf{AND} & \mathbb{P}(\text{A}\,|X > 13) & = & 0.33 \\ \mathbb{P}(\text{A}\,|X \leq 22) & = & 0.75 & \textbf{AND} & \mathbb{P}(\text{A}\,|X > 22) & = & 0.00 \\ \mathbb{P}(\text{A}\,|X \leq 33) & = & 0.60 & \textbf{AND} & \mathbb{P}(\text{A}\,|X > 33) & = & 0.00 \\ \mathbb{P}(\text{A}\,|X \leq 40) & = & 0.50 & \textbf{AND} & \mathbb{P}(\text{A}\,|X > 40) & = & 0.00 \end{array}\right\} \text{12 Cut-point Calculations} \quad (3.3)$$

Computationally, this is very inefficient, as it is extremely costly in time and space complexity. Also, it is likely to be even more expensive with algorithms of the PRISM family, given that they rely on the 'separate-and-conquer' search approach, which may require many iterations to introduce a complete rule.

To address this issue, a new heuristic approach, which is based on Gaussian Probability Density Distribution (GPDD) is proposed in this chapter to generate a more efficient and expressive rule-term structure directly from numeric attributes. Specifically, instead of creating two rule-terms of the form ($\alpha < v$ and $\alpha \geq v$) for every possible value in a numeric attribute (α), only one rule-term of the form ($x \leq \alpha < y$) is induced to describe an interval of data; where v is a discrete value, x and y are valid continuous values of the attribute.

The new Gaussian based rule-term selects only a highly relevant range of values from a numeric attribute α for a given target class. Hence, the 12 cut-point calculations that are described in Rule set 3.3 and used to induce one rule-term for class A, can be replaced by a single rule, which is shown below:

$$\text{if } (9 \leq \ \alpha \leq 22) \text{ Then A} \quad (3.4)$$

This would greatly enhance readability of the individual rules and potentially reduce overfitting. The next section will further explain this method theoretically, while Section 3.7 will evaluate it empirically.

3.5.2 Using Probability Density Distribution to Deal with Continuous Attributes

The idea of utilising the Gaussian probability distribution in the learning process of numeric features is driven by the fact that Gaussian or normal distribution can be considered the most important and most widely used distribution in statistics

and many natural phenomena are at least approximately normally distributed, e.g. human blood pressure, weight, height, age, test scores, etc. [25, 143–146]. Normal distribution is also called the *"Gaussian curve"* after the mathematician 'Karl Friedrich Gauss', however, some studies refer the first discovery of the normal distribution to 'Abraham de Moivre' despite that his theorem about normal distribution was lacked the concept of the probability density function. When the values are normally distributed, the density distribution has a bell shape curve (the height for a given value on the x axis) as shown in Figure 3.1. The parameters μ and σ are the mean and standard deviation, respectively [144].

Furthermore, even if a dataset is not normally distributed, according to 'Central Limit Theorem' (CLT) the distribution of the means of many samples generated from the same dataset would be very nearly normally distributed. Therefore, the more samples generated, the closer the distribution of their means would be to a normal distribution [144]. Also, according to [145], the normality assumption is valid if the sample size is large and the sample are considered to be large enough if its size greater than 30.

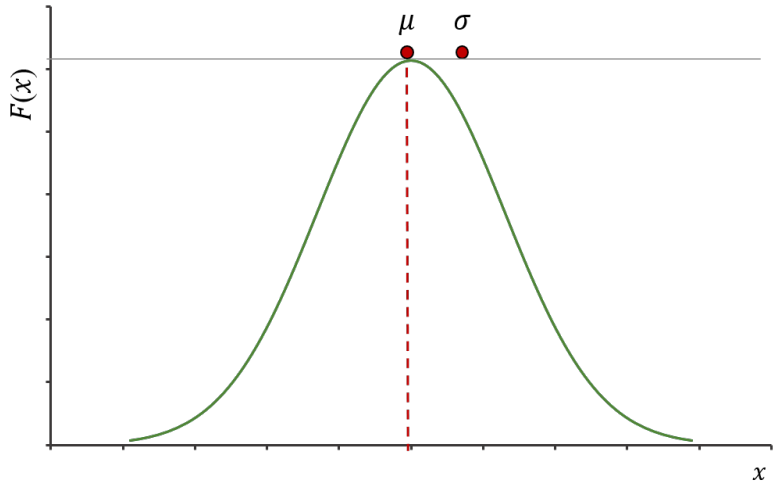


Figure 3.1: Standard normal density function

Assume a dataset with a set of classification labels, $C_1, C_2, C_3, ..., C_i$ and a continuous attribute α. Using Gaussian distribution, we can find the specific value in attribute α that is the most relevant one to a given class label C_i. The Gaussian distribution is calculated with mean μ and variance σ^2 from all the values of the continuous attribute α that are associated with the target class label. Then in order to create a rule-term α_j the conditional density probability for class C_i can be obtained using Equation 3.5.

$$\mathbb{P}(\alpha_j|C_i) = \mathbb{P}((\alpha_j|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(\alpha_j - \mu)^2}{2\sigma^2}) \tag{3.5}$$

Hence, a value based on $\mathbb{P}(C_i|\alpha_j)$ or equivalently $log(\mathbb{P}(C_i|\alpha_j))$ can be calculated using the Equation 3.6. This value is used to determine the posterior probability of a given class label C_i for a valid value of a numerical attribute α_j.

$$log(\mathbb{P}(C_i|\alpha_j)) = log(\mathbb{P}(\alpha_j|C_i)) + log(\mathbb{P}(C_i)) - log(\mathbb{P}(\alpha_j)) \tag{3.6}$$

As shown in Figure 3.2, the greatest probability density is in the middle of the distribution. Thus, from a generated Gaussian probability density distribution (GPDD) for class label C_i, the range of values that extends to both sides from the mean (μ) represent the highly relevant values of the attribute α, i.e. the shaded area under the curve between lower bound (x) and upper bound (y). As a result, a candidate rule-term α_j can be simply produced in the form $(x \leq \alpha < y)$.

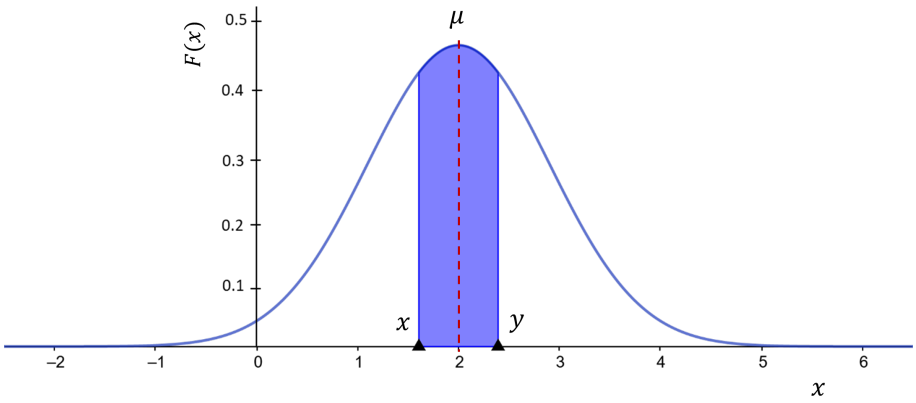


Figure 3.2: The shaded area represents a range of values of attributes α_j for class C_i

Generally speaking, the process of how to generate a rule-term for a target class label using the GPDD based method can be summarised as follows:

1. For each continuous attribute, calculate a Gaussian distribution with mean μ and variance σ^2 from all the values of associated with a given target class.

2. For each numeric value of the current continuous attribute, calculate class conditional density probability using equation 3.5 and then calculate the posterior class probability using 3.6 for the target class.

3. From step 2, determine the mean of the distribution (μ), which is expected to be the attribute's value with the highest class conditional density probability (see Figure 3.2).

4. Select the next lower bound (x) and the next upper bound (y), which represent the next two attribute's values with the highest posterior probabilities to both sides of the mean μ.

5. Create a candidate rule-term in the form $(x \leq \alpha < y)$.

The next section will examine the GPDD based rule-term approach by incorporating it in new modular rule based algorithms and then empirically compare it with the other rule-term structure that is based on frequent cut-point calculations approach.

3.6 G-Prism Algorithms: New Expressive Modular Rule based Algorithms

For the purpose of improving the computational efficiency of PRISM family of algorithms, 'G-Prism algorithms' are developed to produce the more expressive and more efficient rule-terms directly from continuous attributes using the method as introduced in the previous section. The letter 'G' stands for Gaussian probability density distribution. Two versions of G-Prism are proposed in this Chapter: (1) 'G-Prism-FB algorithm' where 'FB' refers to fixed size rule-term boundaries [19], (2) 'G-Prism-DB algorithm' where 'DB' stands for dynamic size rule-term boundaries [20]. Notice that, these algorithms are among the contributions of the research presented in this thesis, and are published in [19, 20]. The following subsections will present each algorithm in depth.

3.6.1 G-Prism Algorithm with Fixed Rule-Term Bounds (G-Prism-FB)

Algorithm 3 illustrates that G-Prism-FB utilises the 'separate and conquer' search strategy. The outer loop iterates over classes, and thus rules are produced for a given class at a time. Each rule is specialised term-by-term by selecting the term that maximise the conditional probability of the rule for a target class. After a complete rule is generated, all instances that covered by that rule are removed from the current training dataset. G-Prism-FB algorithm handles the categorical attributes in exactly the same way as other PRISM family members do, and thus it produces a rule-term α_j of the form $(\alpha = v)$ where v is a discrete value. Concerning continuous attributes, G-Prism-FB can produce a computationally efficient numeric rule-term α_j in the form $(x \leq \alpha < y)$ by using a class conditional density probability of the Gaussian distribution function that has been proposed in Section 3.5.2. x and y are valid continuous attributes values represented by the next closest values left and right of the mean μ, i.e. the shaded area shown in Figure 3.2.

Algorithm 3: Learning classification rules using G-Prism with Fixed Bounds.

Notations: C: class label, D: training dataset, S: subset of D, α: attribute, R: complete rule, v: discrete value, x and y: two numeric values, α_j: attribute-value

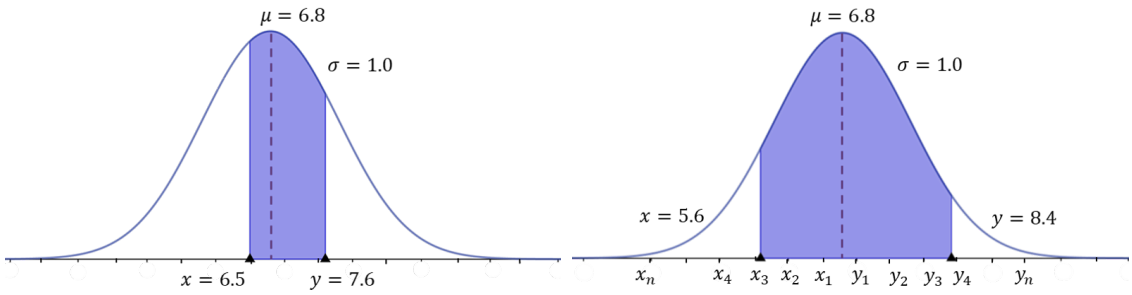
1 **for** $i = 1 \rightarrow C$ **do**
2 $D \leftarrow$ Training Dataset;
3 **while** *D does not contain only instances of class C_i* **do**
4 **forall** *attribute $\alpha \in D$* **do**
5 **if** *attribute α is categorical* **then**
6 Create a rule-term α_j in the form $(\alpha = v)$;
7 Calculate the conditional probability, $\mathbb{P}(C_i|\alpha_j)$ for all possible attribute-value α_j from attribute α ;
8 **else if** *attribute α is continuous* **then**
9 Calculate mean μ and variance σ^2 of continuous attribute α for class C_i ;
10 **foreach** *value α_j of attribute α* **do**
11 Calculate $\mathbb{P}(\alpha_j|C_i)$ based on Gaussian distribution created in line 9 ;
12 **end**
13 Select attribute-value α_j of attribute α, which has highest density in line 11 ;
14 Create a rule-term $(x < \alpha \leq y)$ as explained in Section 3.5.2 ;
15 Calculate $\mathbb{P}(C_i|x < \alpha \leq y)$;
16 **end**
17 **end**
18 Select $(\alpha = v)$ or $(x < \alpha \leq y)$ with the maximum conditional probability (computed in lines 7 and 15) as best rule-term ;
19 Create a subset S from D containing all the instances covered by selected rule-term at line 18 ;
20 $D \leftarrow S$
21 **end**
22 The induced rule R is a conjunction of all selected rule-terms built at line 18 ;
23 Remove all instances covered by rule R from Training Dataset ;
24 **repeat**
25 lines 2 to 23 ;
26 **until** *all instances of class C_i have been removed form the training data*;
27 **Reset** Training Data to its initial state ;
28 **end**
29 **return** *induced Rules* ;

3.6.2 G-Prism Algorithm with Dynamic Rule-Term Bounds (G-Prism-DB)

As explained in the previous subsection, G-Prism-FB makes use of the next smaller attribute value x and the next larger attribute value y from both sides of the mean μ of the Gaussian distribution to find the most common values α_j of continuous attribute α for the target class C_i. Therefore, this algorithm targets the highest density class probability $\mathbb{P}(x \leq \alpha < y|C_i)$ and thus it can generate rules that are computationally less demanding compared with binary splitting approach as demonstrated in Section 3.5.1. However, as it can be seen in Figure 3.3a, G-Prism-FB uses a very conservative strategy for finding the best numeric rule-term boundaries that may lie further left and right of the mean μ than just the next

attribute values. This may result in the rules only covering few instances, which in turn may lead to overfitting of rules and more rules to be induced. The empirical evaluation conducted in this chapter investigates this issue in depth in Section 3.7.

To overcome this problem, the shaded area under the curve can be expanded as shown in Figure 3.3b and thus more attribute values are tested before choosing a highly relevant range of values that maximises the coverage of a rule for a target class. This means that instead of generating one range in the form of $(x < \alpha \leq y)$ by selecting the next lower bound (x) and the next upper bound (y), more ranges can be dynamically produced such as $(x_1 < \alpha \leq y_3)$, $(x_3 < \alpha \leq y_5)$, $(x_2 < \alpha \leq y_4)$,... $(x_n < \alpha \leq y_k)$ or $(x_k < \alpha \leq y_n)$ where k refers to the number of numeric rule-terms to be considered left and right of μ before deciding the best one and n indicates the maximum number of possible rule-terms that can be tested per attribute. Theoretically, the larger k value the larger coverage of data instances for a single rule-term can be found. This new proposed approach, which is named the 'G-Prism with Dynamic Boundaries' (G-Prism-DB) is outlined in Algorithm 4 and empirically evaluated in the next section.



(a) **G-Prism-FB**

$\mathbb{P}(6.5 < \alpha \leq 7.6) = 0.41$

(b) **G-Prism-DB**

$\mathbb{P}(5.6 < \alpha \leq 8.4) = 0.83$

Figure 3.3: Example of finding rule-terms with G-Prism. The shaded areas represent values of attribute α_j for class C_i

The example in Figure 3.3 demonstrates the key difference between the two versions of G-Prism algorithms in terms of the coverage of a particular subset of training instances for a numeric single rule-term. The shaded area in each curve between the vertical lines represents the range of values α_j for a particular class C_i, which are selected to generate a rule-term. In this example with $\mu = 6.8$ and $\sigma^2 = 1.0$, G-Prism-FB generates the rule-term: $(6.5 < \alpha \leq 7.6)$ which covers around 41% of all the possible values under the curve, whereas G-Prism-DB maximises the coverage of the rule-term and thus generates: $(5.6 < \alpha \leq 8.4)$ which covers more than 83% of all the possible values under the curve.

Algorithm 4: Learning classification rules using G-Prism with Dynamic Bounds.

Notations: C: class label, D: training dataset, S: subset of D, α: attribute, R: complete rule, v: discrete value, x and y: two numeric values, α_j: attribute-value, k: number of rule-terms, n: maximum number of possible rule-terms

```
1  for i = 1 → C do
2      D ← Training Dataset;
3      while D does not contain only instances of class C_i do
4          forall attributes α ∈ D do
5              if attribute α is categorical then
6                  Cretae a rule-term α_j in the form (α = v) ;
7                  Calculate the conditional probability, ℙ(C_i|α_j) for all possible
                     attribute-value (α_j) from attribute α ;
8              else if attribute α is numerical then
9                  calculate mean μ and variance σ^2 of continuous attribute α for class C_i ;
10                 foreach value α_j of attribute α do
11                     Calculate ℙ(α_j|C_i) based on Gaussian distribution created in line 9 ;
12                 end
13                 Select attribute-value α_j of attribute α, which has highest density value
                     calculated in line 11 ;
14                 for n = maxBound → 1 do
15                     for k = 1 → maxBound do
16                         Create a rule-term in a form of (x_n < α ≤ y_k) as explained in
                             Section 3.5.2 ;
17                         Calculate ℙ(C_i|x_n < α ≤ y_k) ;
18                     end
19                 end
20             end
21         end
22         Select (α = v) or (x_n < α ≤ y_k) with the maximum conditional probability as a
             rule-term ;
23         Create a subset S from D containing all the instances covered by selected
             rule-term at line 22 ;
24         D ← S
25     end
26     The induced rule R is a conjunction of all selected rule-terms built at line 22 ;
27     Remove all instances covered by rule R from Training Dataset ;
28     repeat
29         lines 2 to 27 ;
30     until all instances of class C_i have been removed form the training data;
31     Reset Training Data to its initial state ;
32 end
33 return induced Rules ;
```

G-Prism-DB algorithm, as illustrated in Algorithm 4 (line 14), allows the user to specify the maximum upper and lower bound based on μ and σ^2, and thus any combination of numeric rule-terms within these bounds is taken into account. For example, if the user has chosen a bound of 3, then there are (3^2) possible rule-terms that are need to be examined for this attribute before deciding the best one. Nevertheless, the ideal boundary may lie beyond this predefined value. Theoretically, it is possible not to impose a bound and allow any possible rule-term combinations fanning out from μ. However, this is likely to result in a computa-

tionally very expensive approach, especially if there are many distinct values for a particular attribute. Also, statistically the larger the number of distinct values, the more likely it is that the maximum boundary is close to μ and therefore it is unlikely that rule-terms spanning far from the μ will cover many instances of the target class.

Furthermore, as the algorithm follows a 'separate and conquer' strategy, the number of training instances decreases over time. Thus, after each iteration μ and σ^2 are updated and the bounds are selected from the currently available values of the continuous attribute for the current target class. Also, like other PRISM family members, G-Prism-DB has the ability to abstain from making a classification when it is uncertain. Regarding the categorical attributes, G-Prism-DB uses the PRISM rule induction strategy to induce the discrete rule-terms as it can be seen in Algorithm 4 (lines 5 - 7).

3.7 Empirical Evaluation of G-Prism Algorithms

The aim of the experimental evaluation, in this chapter, is to compare the performance of the dynamic rule-term boundary in G-Prism-DB algorithm with the binary splitting in PRISM and the fixed rule-term boundary in G-Prism-FB algorithm. As previously explained in Section 3.5.1, binary splitting is the local discretisation method that has been applied to the PRISM family to deal with numerical attributes as outlined in [5, 69]. Therefore, the comparison presented here is against an implementation of PRISM that incorprates the aforementioned extensions of PRISM Family members. This implementation is referred here as PRISM.

3.7.1 Experimental Setup

All the three algorithms have been implemented in the statistical programming language R [147][2]. The fixed sized boundary of G-Prism-FB was set to one value smaller and one value larger than μ. The dynamic sized boundary originally allows the user to determine the lower and upper boundary. However, the current experiment was set to allow a range up to 6 smaller and 6 larger values based on μ and σ^2. The algorithms have been applied to 11 datasets, which are described in Table 3.2. These datasets were picked randomly from the UCI repository [17] and the only condition being that they contain continuous attributes and involve classification tasks. The reason for choosing datasets with continuous features

[2]All the source codes are available in a public online repository at `https://github.com/ManalAlmutairi/PhD_Project_Codes/tree/v1.0.0` and are archived at `https://doi.org/10.5281/zenodo.5557590` [23].

only is because the G-Prism-FB and G-Prism-DB algorithms are distinguished from the original PRISM only with regard to dealing with continuous attributes. The datasets have been randomly sampled without replacement into train and test datasets. While the test set consists of 30% of the dataset, the remaining 70% were used to build the classifiers. Please note that the same experiments on the same datasets plus additional datasets were conducted within a comparative evaluation in Chapter 5. The overal results for G-Prism versions were the same.

Table 3.2: List of Datasets used in the experiments

	Dataset	No. Instances	No. Attributes	No. Classes
1.	iris	150	4 (cont)	3
2.	seeds	210	7 (cont)	3
3.	wine	178	13 (cont)	3
4.	blood transfusion	748	5 (cont)	2
5.	banknote	1372	5 (cont)	2
6.	ecoli	336	8 (7 cont, 1 name)	8
7.	yeast	1484	9 (8 cont, 1 name)	10
8.	page blocks	5473	10 (cont)	5
9.	user modelling	403	5 (cont)	4
10.	breast tissue	106	10 (cont)	6
11.	glass	214	10 (9 cont, 1 id)	7

The algorithms were evaluated on each of the datasets against the following 5 evaluation metrics:

- *Number of rules*: this is simply the total number of rules induced. A low number of rules is desired.
- *Abstaining Rate*: PRISM, G-Prism-FB, G-Prism-DB algorithms abstain from a classification if a case is not covered in the rule set. The abstain rate is the ratio of instances that remain unclassified in the test set. A low abstain rate may be desired.
- *F1 score*: this is the harmonic mean of precision and recall. A high F1 score is desired.
- *Accuracy*: this is the ratio of data that have been correctly classified. Instances that not covered in the rule set are classified using the majority class strategy. A high accuracy is desired.
- *Tentative Accuracy*: this is the ratio of data that have been correctly classified among only the classification attempts. A high tentative accuracy is desired.

It is worth noting that there is a direct relationship between accuracy, tentative accuracy and abstaining rate. The accuracy counts abstained instances as a potential misclassification, because the majority class label in a dataset may not always be

the correct prediction for all the abstained instances. Tentative accuracy does not consider abstained instances. Therefore, the higher the abstaining rate, the lower the accuracy and the higher the tentative accuracy.

3.7.2 Empirical Results

Table 3.3 and Table 3.4 show the results of the experiments with respect to the 5 evaluation metrics. In each table, the '#' symbol refers to the index of the dataset in Table 3.2. Regarding the number of Rules, Table 3.3 shows PRISM produces fewer rules than G-Prism algorithms in 7 out of 11 datasets. However, comparing both versions of G-Prism, G-Prism-DB is outperformed in all the 11 cases. In terms of abstaining rate, which is the ratio of instances that remain unclassified in the test set, Figure 3.4 shows that PRISM seems to have a lower abstaining rate in most cases (9 out of 11). In comparison with G-Prism-FB, G-Prism-DB achieves a lower abstaining rate in 7 out of 11 cases.

Table 3.3: Results of Number of Rules and Abstaining Rates

Datasets #	Number of Rules			Abstaining Rate		
	Prism	G-Prism		Prism	G-Prism	
		FB	DB		FB	DB
1	12	20	**9**	0.04	**0.02**	0.04
2	**19**	73	30	**0.02**	0.06	0.10
3	**12**	55	16	**0.06**	0.19	0.11
4	**19**	109	46	**0.00**	0.32	0.10
5	**13**	466	176	**0.00**	0.09	0.08
6	51	113	**49**	**0.04**	0.28	0.21
7	**78**	595	287	**0.00**	0.28	0.12
8	**138**	1236	465	**0.00**	0.03	0.03
9	**26**	122	41	0.29	0.28	**0.17**
10	31	42	**23**	**0.13**	0.44	0.31
11	54	77	**46**	**0.09**	0.46	0.48

Table 3.4: Results of F1 score, Overall Accuracy and Tentative Accuracy

Datasets #	F1 score			Overall Accuracy			Tentative Accuracy		
	Prism	G-Prism		Prism	G-Prism		Prism	G-Prism	
		FB	DB		FB	DB		FB	DB
1	0.90	**0.93**	**0.93**	0.87	**0.91**	**0.91**	0.91	**0.93**	**0.93**
2	0.88	**0.94**	0.93	**0.87**	**0.87**	0.86	0.87	**0.93**	**0.93**
3	**0.98**	0.89	0.96	**0.92**	0.79	0.89	**0.98**	0.88	0.96
4	0.87	**0.90**	0.89	0.76	**0.77**	**0.77**	0.76	**0.82**	0.81
5	0.80	**0.96**	**0.96**	0.72	0.90	**0.92**	0.72	0.95	**0.96**
6	0.37	0.69	**0.76**	0.67	0.65	**0.75**	0.68	0.84	**0.88**
7	0.29	0.44	**0.45**	0.37	0.40	**0.43**	0.36	0.47	**0.48**
8	0.58	**0.82**	0.79	**0.95**	**0.95**	**0.95**	0.95	**0.96**	**0.96**
9	0.66	0.84	**0.92**	0.53	0.66	**0.78**	0.72	0.83	**0.91**
10	0.71	0.79	**0.93**	**0.66**	0.50	**0.66**	0.75	0.89	**0.95**
11	0.55	0.44	**0.73**	0.52	0.51	**0.66**	0.53	0.54	**0.82**

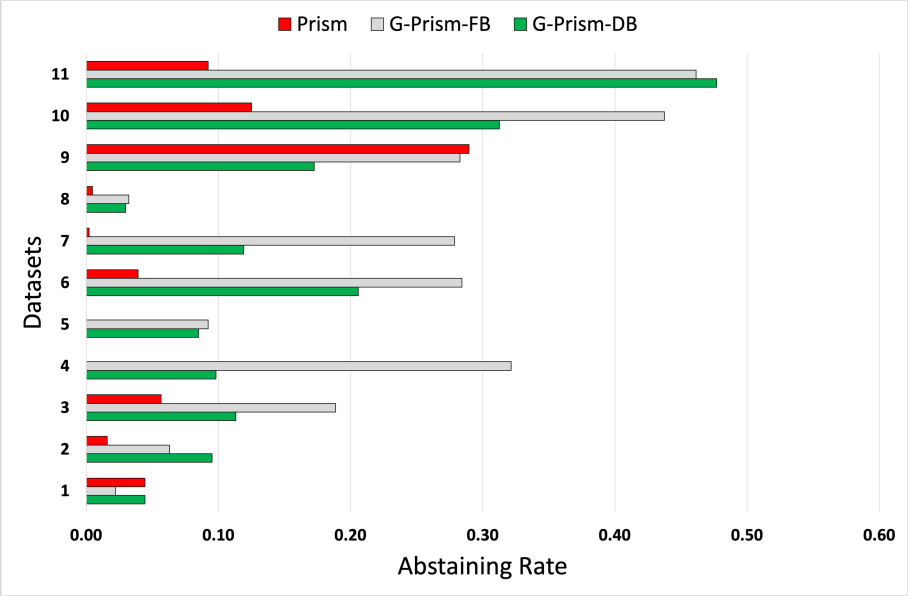


Figure 3.4: Abstaining rates of PRISM, G-Prism-FB and G-Prism-DB

Moreover, Table 3.4 lists the results for the remaining evaluation metrics; F1 score, accuracy and tentative accuracy. For easier description of the contents of the table, Figure 3.5 demonstrates the difference of the F1 score of G-Prism-FB and G-Prism-DB compared with the basic PRISM, Figure 3.6 illustrates the difference of the accuracy of G-Prism-FB and G-Prism-DB compared with the basic PRISM, and Figure 3.7 presents the difference of the tentative accuracy of G-Prism-FB and G-Prism-DB compared with the basic PRISM.

With respect to F1 score, Figure 3.5 shows that G-Prism algorithms outperforms PRISM in most cases (10 out of 11) and G-Prism-DB in particular achieves the higher score in 7 out of these 10 cases compared with G-Prism-FB.

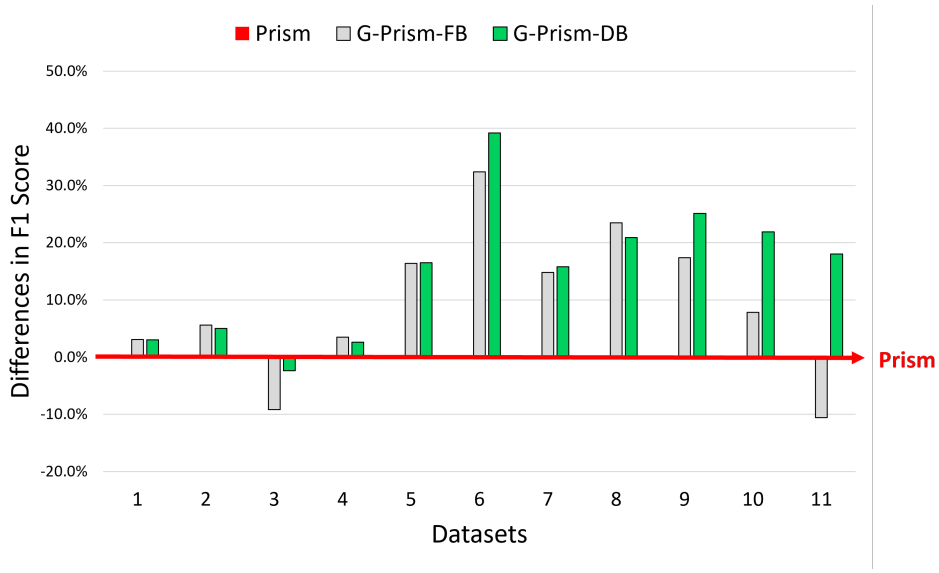


Figure 3.5: Difference of F1 score of G-Prism-FB and G-Prism-DB compared with Prism

Regarding overall accuracy, as can be seen in Figure 3.6, G-Prism-DB achieves the better accuracy compared with PRISM in 7 out of 11 datasets. However, in the cases where G-Prism-DB did not outperform PRISM, its accuracy only marginally lower. Comparing with PRISM, G-Prism-FB has a higher accuracy in 5 datasets and in 4 it does not; however, it performs at the same level with PRISM at 2 cases. In terms of tentative accuracy, as can be seen in Figure 3.7, both versions of G-Prism outperform PRISM, both achieve a higher tentative accuracy in 10 out of 11 datasets. Also, G-Prism-DB achieves a better accuracy in 7 out of 11 cases in comparison with G-Prism-FB. In all cases where G-Prism-DB did not perform better than G-Prism-FB, it achieved only marginally lower tentative accuracy.

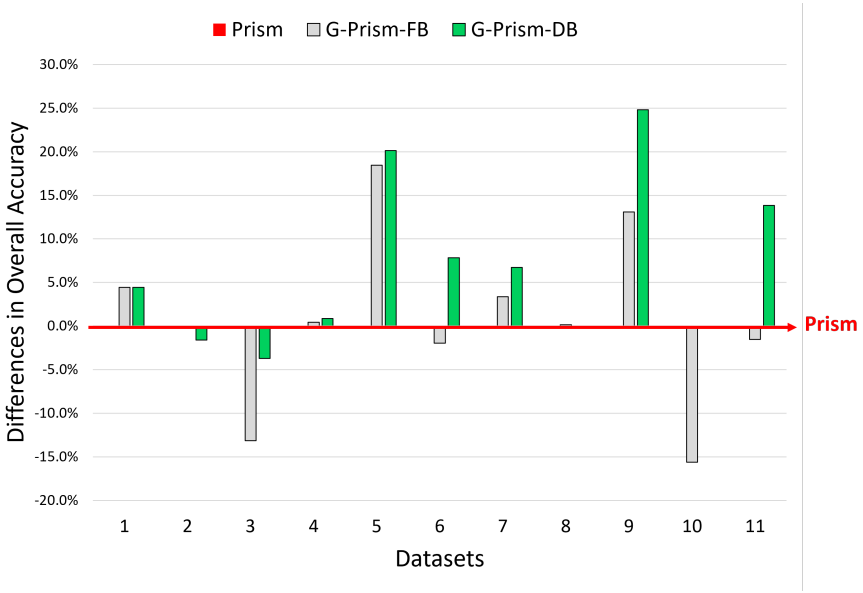


Figure 3.6: Differences in overall Accuracy of G-Prism-FB and G-Prism-DB compared with Prism

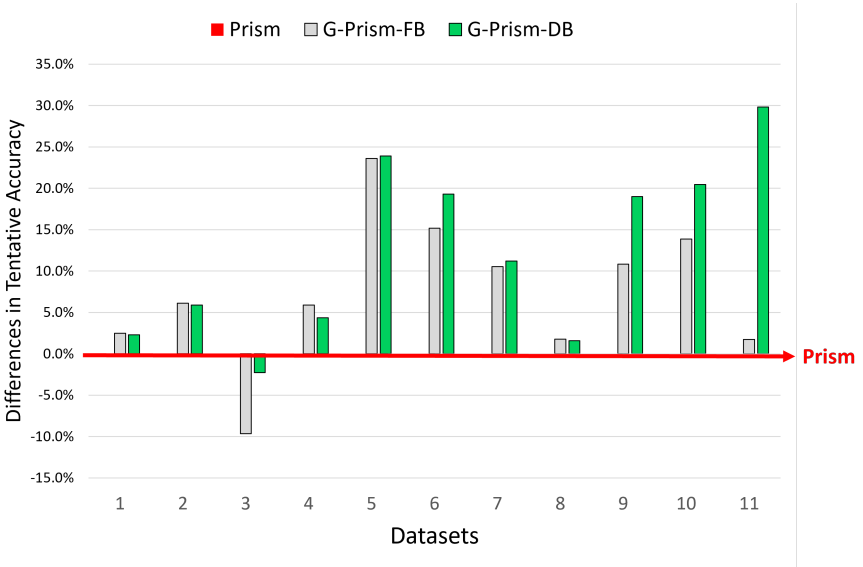


Figure 3.7: Differences in Tentative Accuracy of G-Prism-FB and G-Prism-DB compared with Prism

3.7.3 Interpretation of Results

Overall, comparing with the original PRISM algorithm, it can be seen that both versions of G-Prism algorithm are outperformed in most cases, in terms of the three evaluation metrics (F1 score, overall accuracy, and tentative accuracy). However, with regards to the number of rules and abstaining rates, PRISM outperforms G-Prism algorithms in most cases. The high abstaining rates in G-Prism approaches could potentially link to the high number of rules induced by G-Prism. In other words, a large number of rules with little coverage of each rule means that the model might have overfitted the training data and hence fewer test data can be covered by the rule set.

Comparing the two approaches of G-Prism with each other, the main difference between them as previously illustrated in Figure 3.3 is that G-Prism-DB had a dynamic rule-terms boundaries that are either the same or cover a wider range. Therefore, in all cases, G-Prism-DB produces fewer rules but covering a larger number of training instances compared with G-Prism-FB. Also, in terms of the remaining measurements, all the figures in this section show that G-Prism-DB algorithm outperforms G-Prism-FB in most cases. Generally speaking, G-Prism-DB achieves a better classification performance compared with Prism and G-Prism-FB. However, more improvements are required to reduce the number of rules comparing with the basic Prism. Moreover, some limitations that exist in both G-Prism algorithms will be highlighted and addressed in the next chapter.

3.8 Summary

This chapter has discussed a number of limitations that exist in separate and conquer approaches, namely PRISM family of algorithms. It has focused specifically on the challenges in dealing with numeric attributes. A common way to handle this issue is to convert numerical features into categorical ones, i.e. known as 'discretisation'. Therefore, Section 3.4 has thoroughly reviewed a number of discretisation methods, which have been categorised into local and global. Local discretisation such as cut-point calculations and global discretisation such as ChiMerge and CAIM methods are illustrated in Section 3.4.1. The computational issues with discretising continuous attributes using cut-points calculations in PRISM family of algorithms are investigated in Section 3.5.1.

Subsequently, a new computationally efficient approach of constructing numerical rule-terms in the form ($x \leq \alpha < y$) instead of two separate rule-terms combinations was proposed in Section 3.5.2. The new rule-term structure is based on Gaussian Probability Density Distribution (GPDD) and it has been utilised to

develop two new members in PRISM family of algorithms; termed G-Prism-FB and G-Prism-DB, which were proposed in Section 3.6.1 and Section 3.6.2 respectively. These two algorithms are among the contributions of this project and they were published in [19, 20].

The main difference between these two algorithms as illustrated in Figure 3.3 is that G-Prism-DB had a dynamic rule-terms boundaries that can be set by the user to expand the coverage of each rule and thus cover a larger number of instances. An empirical performance evaluation of G-Prism algorithms and basic PRISM is detailed in Section 3.7. The results show that the classification performance of G-Prism-DB algorithm is better than PRISM and G-Prism-FB in most cases with regard to these 3 metrics: F1 score, accuracy and tentative accuracy. In terms of abstaining rate and number of rules induced, PRISM algorithm performs better than both G-Prism approaches in most cases. Therefore, an improvement to the results can be made to the more accurate algorithm in this chapter (G-Prism-DB) to reduce the number of rules induced. The following chapter will investigate and address this issues, in addition to a number of other limitations that exist in G-Prism approaches.

Chapter 4

G-Rules-IQR: a Classifier with Expressive, Explainable and Accurate Rule-Terms

This chapter introduces a new rule-based algorithm, called G-Rules-IQR, which is based on a combination of GPDD approach[1], Interquartile Range (IQR), and a transformation approach towards normally distributed data. The chapter also, illustrates the methods that have been integrated in G-Rules-IQR theoretically and empirically. Please note that G-Rules-IQR algorithm is among the contributions of the research presented in this thesis, and it is published in [21].

4.1 Introduction

The previous chapter introduced a new method based on the density class probability from Gaussian distribution to extract numerical rule-terms directly from continuous attributes. This new rule-term structure was integrated into two new rule-based algorithms; termed G-Prism-FB and G-Prism-DB. The algorithms are resulted in a better accuracy compared with the original Prism. Comparing both version of G-Prism, G-Prism-DB generally achieves a better classification performance compared with G-Prism-FB. However, it is possible that this approach could be improved. As a result, the following research objective, which was partially met in the previous chapter is fully met in the current chapter:

[1]GPDD stands for Gauss Probability Density Distribution which is presented in Chapter 3 (see Section 3.5.2)

> **Objective 2:** *"To measure and compare the expressiveness of rule based models and develop an appropriate rule-based predictive algorithm suitable as base learner for an ensemble."*

Measuring the expressiveness of a rule-based learner often depends on the complexity of its rule set. A rule set is considered more expressive when it produces fewer number of rules with less complex terms per rule.

This chapter highlights some problems that exist in G-Prism algorithms. The solutions resulting in a new rule induction classifier, termed '**G-Rules-IQR**' are presented. The proposed algorithm is based on quartiles and Interquartile Range (IQR) to derive the upper and lower boundaries of rule-terms. This is introduced in Section 4.5 and evaluated against three different implementations of the original Prism and the two versions of G-Prism.

The main advantage of this approach is that the generated rule-terms are more expressive and computationally less demanding compared with the local discretisation approach (binary splitting) in PRISM family and dynamic boundaries approach in G-Prism. Therefore, G-Rules-IQR classifier is expected to be more appropriate to be used as base classifier to the ensemble system that can sufficiently meet the final goal of this thesis, which is to develop a predictive ensemble learner that is human-readable (expressive) while retaining the key advantage of ensembles which are the better accuracy than its stand-alone base classifiers.

Moreover, in Sections (4.3, 4.4 and 4.5.1), the chapter explains the methods utilised or developed to overcome the aforementioned shortcomings of G-Prism algorithms and then incorporated in G-Rules-IQR as shown in Figure 4.4. Lastly, a short summary of this chapter is provided.

4.2 Limitations in G-Prism Algorithms

As previously discussed, G-Prism-FB and G-Prism-DB algorithms use a new heuristic rule-term induction method, which handles the continuous attributes more efficiently than original PRISM algorithm. However, there are 4 limitations of these algorithms that can be highlighted as follows:

1. *User defined threshold.* Regarding the more accurate G-Prism-DB classifier, the user has to define the maximum number rule-term boundary values to the left and right of μ (by default, six values to the left and six values to the right). However, the optimal boundary may lie beyond this user defined value and is dependent on the number of training instances. Statistically, the larger the number of training instances, the more likely that the ideal boundaries are closer to mean, i.e. the need to define a large number rule-term

boundary values becomes less important. Nevertheless, being constraint by an external parameter takes from the user can be considered an obstacle.

2. *Normal Distribution Assumption.* Although this assumption is common in many statistical procedures and models [148], it may not be accepted in some applications and thus considered a drawback. The experiments of both versions of G-Prism algorithms described in the previous chapter (Section 3.7) did not test the attributes' distributions in the experiments. Despite that the results reflect a good performance for G-Prism-DB algorithm in most cases, it is still possible that the algorithm may not perform as well on attributes that are not normally distributed due to the use of GPDD. Normality assumption in classification tasks will be further discussed in Section 4.4

3. *Abstaining rate.* The experiments conducted in the previous chapter show that the abstaining rates for both approaches of G-Prism are higher than the one of original Prism. This was the reason for decreasing the accuracy of the classifiers in some cases despite the higher tentative accuracy values. This is because abstained instances were counted as misclassification.

4. *Execution time:* Although this evaluation metric not has been used in the empirical evaluation detailed in Section 3.7, it is expected that Prism and G-Prism-DB are computationally more expensive than G-Prism-FB as a result of additional and frequent cut-point calculations for Prism and multiple additional rule-term bounds evaluations for G-Prism-DB.

Regarding problem (1) the user defined rule-term boundary threshold, which is required by G-Prism-DB algorithm, this chapter proposes a new rule-term structure based on Interquartile Range (IQR) to addresses this issue. Statistically, IQR is a popular way to measure the dispersion (spread) of a numerical dataset. The next section illustrates the concept of IQR and its advantages of using it to induce rule-terms from continuous attributes.

In terms of problem (2) the normality distribution assumption, a solution to overcome this obstacle is proposed in Section 4.4. With respect to obstacles (3) and (4), they will be investigated in depth in the evaluation part of this chapter (Section 4.6).

4.3 Measuring the Dispersion of Numerical attributes

The degree to which numerical data tend to spread is called the dispersion or variability. The most widely used measures of data dispersion are standard deviation, variance, quartiles, and IQR. The dispersion measures refer to how often

and how much they differ from the mean μ [1]. Variance is the most commonly used dispersion measure. The standard deviation is the square root of the variance. Quartiles as shown in Figure 4.1 are the values that divide the probability density function into four parts with an equal amount of data points (25% each). The second quartile (Q2) is identical to the median. The lower quartile, denoted by Q1 in the figure, is the 25^{th} percentile; and the upper quartile, denoted by Q3, is the 75^{th} percentile. The IQR represents the range of attribute values that cover the middle half of the dataset, which is equal to $Q3 - Q1$. While the standard deviation and variance are very sensitive to a single large or small value (outliers), quartile and IQR are robust to outliers. Also, IQR is very useful for describing skewed distributions [1].

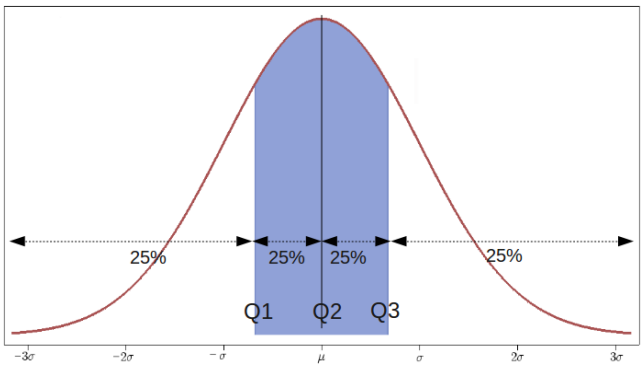


Figure 4.1: Interquartile Range (IQR) of normal random variables

Advantages of using IQR Over the Mean and Variance to Measure the Spread of the Dataset:

The two examples shown in Figure 4.2 and Figure 4.3 demonstrate the robustness of the IQR based method against the outliers in comparison to the variance based method. Assume that the continuous attribute values that are shown in each figure are under the Gaussian curve, and they are all relevant to the same target class. Notice that the attribute's values in Figure 4.3 include an extreme outlier value (400). In both figures, the numbers indicated by the letter **(a)** represent the upper and lower boundaries that can be extracted using the mean and variance, while **(b)** indicates the upper and lower boundaries that can be derived using IQR.

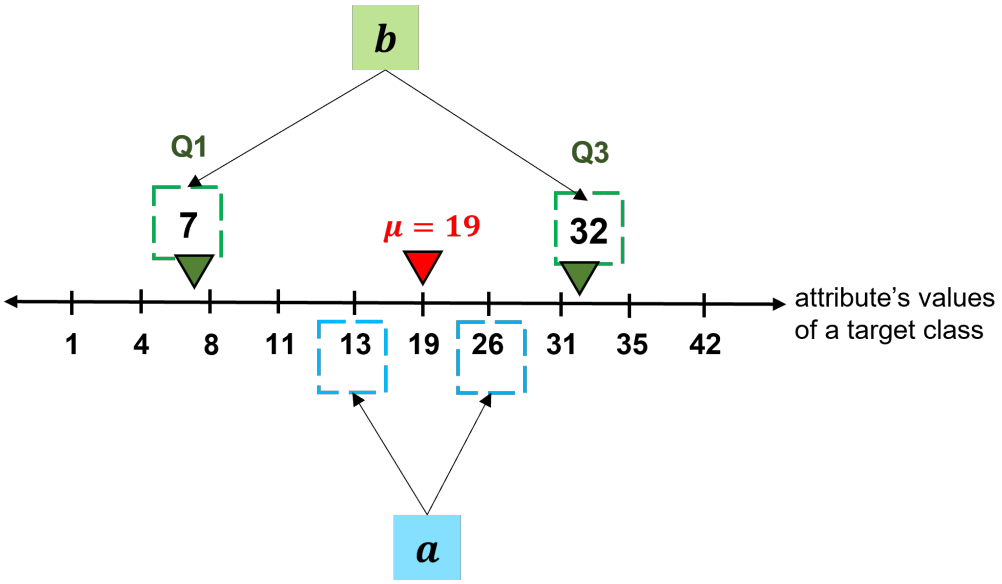


Figure 4.2: An example of inducing a numerical rule-term from an attribute's values of a target class. **(a)** Rule-term generated using mean and variance; **(b)** Rule-term generated using IQR

From the attribute's values presented in Figure 4.2, the following values can be computed:

$$\left.\begin{array}{l}\text{Mean}(\mu) = 19\\ \text{first quartile (Q1)} = 7\\ \text{second quartile (Median)} = 16\\ \text{Third quartile (Q3)} = 32\\ \text{Interquartile Range (IQR)} = 25\end{array}\right\} \quad \text{measures of spread}$$

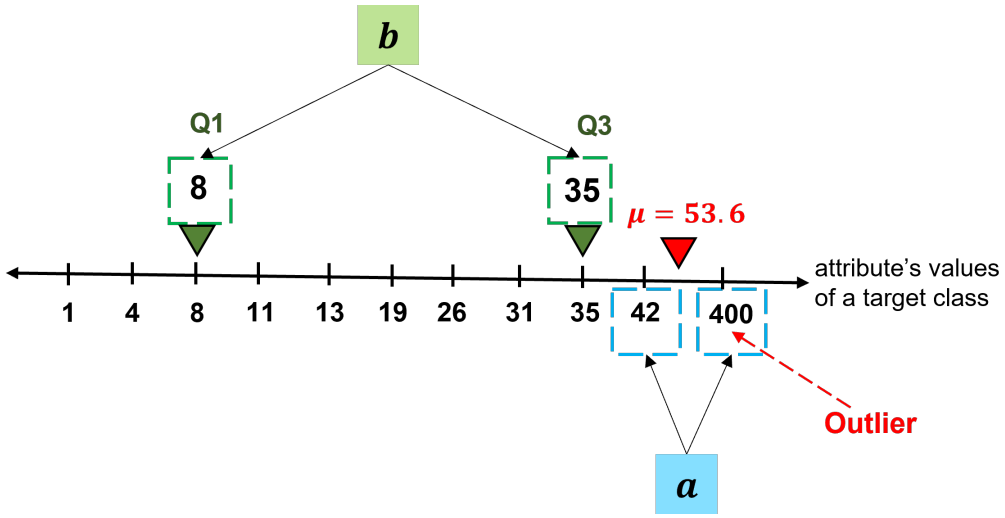


Figure 4.3: An example of inducing a numerical rule-term from an attribute's values of a target class with an extreme outlier added to it. **(a)** Rule-term generated using mean and variance; **(b)** Rule-term generated using IQR

From the attribute's values that contained an extreme outlier value (400), shown in Figure 4.3, the following values can be computed:

$$\left.\begin{array}{l}\text{Mean}(\mu) = 53.63\\ \text{first quartile (Q1)} = 8\\ \text{second quartile (Median)} = 19\\ \text{Third quartile (Q3)} = 35\\ \text{Interquartile Range (IQR)} = 27\end{array}\right\} \quad \text{measures of spread}$$

Comparing the measures of spread results demonstrates that the IQR based method is not effected by outliers as the IQR value only slightly changed from 25 to 27, and thus Q1 and Q3 locations are almost unchanged. However, the mean has been considerably effected by the outlier as its value changed from 19 to 53.63 and thus its location has moved a lot toward the outlier.

4.4 Assumption of Normality

Assuming that the underlying data is normally distributed is one of the most common assumptions made in the development and use of statistical procedures and models. It has a considerable attention in the literature and therefore there are more tests designed specifically to assess normality than for any other distribution [148]. Generally speaking, the assumption is based on a number of Central Limit Theorem (CLT) characteristics, which can be summarised as follows:

- The distribution of a random sample generated from a dataset would be very nearly normally distributed if the sample size is large enough, even if the dataset itself is not normally distributed.
- If a random sample derived from normal distributed data is drawn, then the distribution of the sample is normal distributed regardless of its size.
- The distribution of means of many random samples generated from the same dataset with any distribution would tend to be normally distributed.

Furthermore, the experiments conducted in [149, 150] conclude that the violation of the normality is not a major issue when we have samples of hundreds of observations. Because of the above points, the approaches developed in the previous chapter (G-Prism algorithms) are dealing with continuous attributes based on the assumption that the values come from a normal or Gaussian distribution. However, if this assumption is not justified, then Section 4.5.1 proposes an approximation method toward normally distributed data, which can overcome or

considerably mitigate this limitation. The approach utilises a prior normality test to decide whether the assumption is at least approximately satisfied or not. The next section introduces the methods adopted in this research for normality test.

Normality Tests:

There are two main methods of assessing normality: Graphical and numerical. Graphical or visual methods might be useful when sample sizes are very small or very large. However, to avoid the wrong interpretations of the graphical tests, it might be the best to rely on the numerical methods [150].

There are several numerical approaches available to test the normality of continuous attributes, among them are the two widely used: Shapiro-Wilk test [151] and Jarque–Bera test [152]. The former is based on the correlation between the data and the corresponding normal scores, and the latter is based on the function of the measures of skewness S and kurtosis K compared from the sample [153]. Under normality, the values of S and K in Jarque-Bera test are 0 and 3 respectively. According to [150, 153, 154], Shapiro-Wilk test is more suitable method for small sample sizes (< 50 samples) whereas Jarque-Bera become more powerful while the sample size increases.Noticeably, the sizes of all the datasets used in the experimental studies in this thesis that have been conducted in the current project are larger than 50 instances. Therefore, Jarque-Bera is chosen in this thesis for testing the normality of data before using the new rule-based algorithm that will be proposed in the next section.

4.5 G-Rules-IQR Algorithm

G-Rules-IQR, a rule-based algorithm, was developed with the aim of overcoming or mitigating some limitations and drawbacks of both versions of G-Prism algorithm that have been listed in Section 4.2. The new proposed approach is presented in Algorithm 5 and is expected to be not only expressive but also more accurate and fast compared with its predecessors in order to fully meet objective 2. The following subsections describe the procedures incorporated in G-Rules-IQR algorithm, which are summarised in Figure 4.4.

4.5.1 Transformation for Skewed Distribution

One of the major limitations of G-Prism-FB and G-Prism-DB algorithms is the assumption of normally distributed attributes. In order to address this obstacle, G-Rules-IQR algorithm integrates a testing for normality for each attribute in the

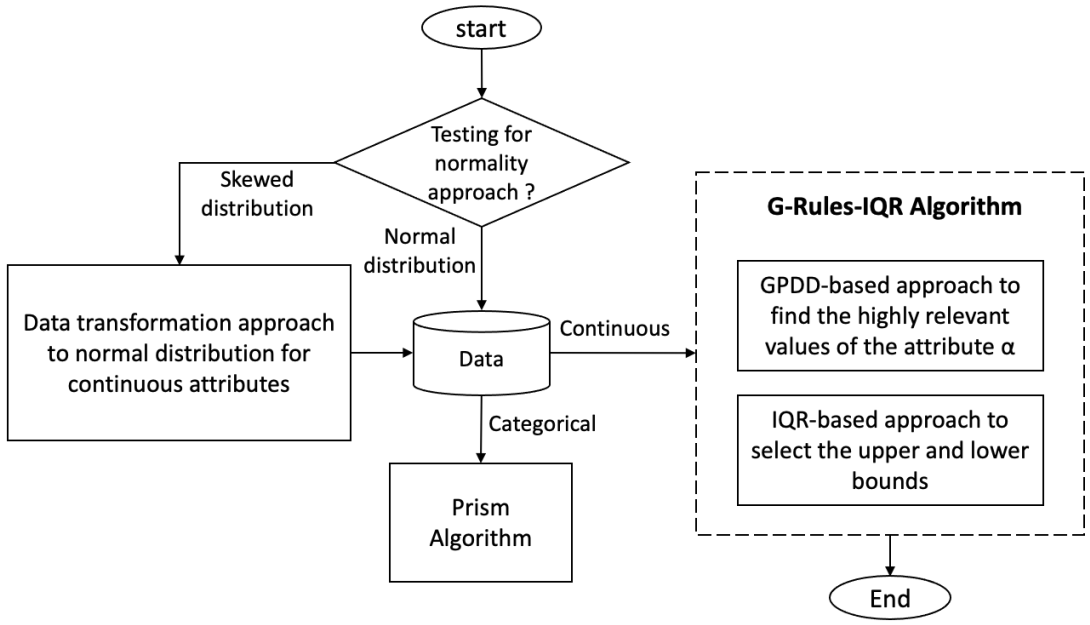


Figure 4.4: The main approaches incorporated in G-Rules-IQR algorithm

dataset as shown in Figure 4.4. The test is carried out prior to the application of G-Rules-IQR algorithm using the widely used normality test, Jarque-Bera. Hence, if values of an attribute at a particular target class are not normally distributed, then the algorithm would apply an approximate normal transformation of the attribute's values with respect to that target class. In other words, it reduces the skewness of attribute values from the normal distribution. A simple and common method to achieve the transformation for a skewed long-tailed dataset is to take the logarithm of the skewed attribute values [148]. This method of approximation to normal distribution is utilised in this research.

4.5.2 A new approach to induce numerical rule-terms using Quartiles and Interquartile Range (IQR)

It can be seen from Figure 4.4 that the proposed G-Rules-IQR algorithm uses the GPDD-based approach, which was introduced in the previous chapter (Section 3.5.2) to find the most relevant values of an attribute for a target class. In this approach, as shown in Algorithm 5, the Gaussian distribution is calculated for each continuous attribute α from all the values associated with a classification C_i. Then, the class conditional density probability for each numeric attribute-value α_j of the target class C_i is calculated (line 11). The attribute-value α_j, which has the highest density is then selected. To create a candidate rule-term from α_j in the form $(x \leq \alpha < y)$, G-Rules-IQR incorporates the quartiles method, which divides the probability density function into four parts with an equal amount of

data points (25% each). The main idea of the method was previously explained in Section 4.3 (Figure 4.1), and the size of its coverage is dependent on the size of the dataset. Specifically, G-Rules-IQR makes use of the difference between the third and the first quartiles as in Equations 4.1 to find the upper rule-term and the lower rule-term boundaries. σ is the standard deviation from the mean, z_1 is the fixed score of the first quartile and is (≈ -0.67), while z_3 is the fixed score of the third quartile and is (≈ 0.67).

$$\begin{aligned} Q_1 &= (\sigma * z_1) + x \\ Q_3 &= (\sigma * z_3) + x \\ IQR &= Q_3 - Q_1 \end{aligned} \tag{4.1}$$

Algorithm 5: Learning classification rules using G-Rules-IQR Algorithm

Notations: C: class, D: training dataset, S: subset of D, α: attribute, R: complete rule, v: discrete value, x , y: two numeric values, α_j: attribute-value

```
1  for i = 1 → C do
2      D ← Training Dataset;
3      while D does not contain only instances of class C_i do
4          forall attributes α ∈ D do
5              if attribute α is categorical then
6                  Create a rule-term α_j in the form (α = v) ;
7                  Calculate the conditional probability, ℙ(C_i|α_j) for all possible attribute-value (α_j) from
                     attribute α ;
8              else if attribute α is continuous then
9                  Calculate mean μ and variance σ^2 of continuous attribute α for class C_i ;
10                 foreach value α_j of attribute α do
11                     Calculate ℙ(α_j|C_i) based on Gaussian distribution created in line 9 ;
12                 end
13                 Select attribute-value α_j of attribute α which has highest density in line 11 ;
14                 Compute 1st and 3rd quartile using zscore values ;
15                 zScore = 0.67 ;
16                 x = σ * (−zScore) + α_j ;
17                 y = σ * (zScore) + α_j ;
18                 Create a rule-term (x < α ≤ y) ;
19                 Calculate ℙ(C_i|x < α ≤ y) ;
20             end
21         end
22         Select (α = v) or (x < α ≤ y) with the maximum conditional probability (computed in lines 7 and 19)
             as best rule-term ;
23         Create a subset S from D containing all the instances covered by selected rule-term at line 22 ;
24         D ← S
25     end
26     The induced rule R is a conjunction of all selected rule-terms built at line 22 ;
27     Remove all instances covered by rule R from Training Dataset ;
28     repeat
29         lines 2 to 27 ;
30     until all instances of class C_i have been removed form the training data;
31     Reset Training Data to its initial state ;
32 end
33 return induced Rules ;
```

4.6 Comparative Experimental Evaluation of G-Rules-IQR Algorithm

The aims of the experiments in this chapter are:

- To evaluate the performance of G-Rules-IQR algorithm compared with its predecessors G-Prism-FB and G-Prism-DB. Unless stated otherwise, the default parameters of these algorithms as stated in Section 3.6 have been used. The implementation of G-Rules-IQR allowed to switch off the transformation to approximate normal distribution.
- To compare G-Rules-IQR and the G-Prism algorithms with original PRISM ALGORITHM using three different discretisation methods to handle the continuous attributes indirectly. Further explanations about these versions of PRISM are given in Section 4.6.2.

4.6.1 Experimental Setup

All the experiments were performed on a $2.3\ GHz$ Intel Core $i7$ machine with 16 GB DDR3 memory, running macOS High Sierra version $10.13.2$. The procedure used in this experimental evaluation is hold-out procedure. All the 18 datasets used in the experiments were picked randomly from the UCI repository [17], the only condition being that they contain continuous attributes and involve classification tasks.

Table 4.1: List of Datasets used in the experimental evaluation of G-Rules-IQR algorithm

#	Dataset	No. Attributes	No. Classes	No. Instances
1.	iris	4 (cont)	3	150
2.	seeds	7 (cont)	3	210
3.	wine	13 (cont)	3	178
4.	blood transfusion	5 (cont)	2	748
5.	banknote	5 (cont)	2	1372
6.	ecoli	8 (7 cont, 1 name)	8	336
7.	yeast	9 (8 cont, 1 name)	10	1484
8.	page blocks	10 (cont)	5	5473
9.	user modelling	5 (cont)	4	403
10.	breast tissue	10 (cont)	6	106
11.	glass	10 (9 cont, 1 id)	7	214
12.	HTRU2	9 (cont)	2	17898
13.	magic Gamma	11 (cont)	2	19020
14.	wine quality-white	12 (cont)	11	4898
15.	letter recognition	17 (cont)	26	20000
16.	breast cancer	11 (10 cont, 1 id)	2	699
17.	post-operative	9 (8 categ, 1 cont)	3	90
18.	EEG eye state	15 (cont)	2	14980

All algorithms have been implemented in the statistical programming language R [147] and re-used the same code base, differing only in the methodological aspects described in this chapter. The datasets have been randomly sampled without replacement into train and test datasets; whereas the test set consists of 30% the dataset and the remaining 70% was used to learn the ruleset. The datasets are described in Table 4.1 in terms of number of instances, attributes (and type of attributes) and classes. Datasets 16 and 17 contained missing values. Missing categorical values have been replaced with the most frequent categorical value for the concerning attribute, and missing continuous values have been replaced with the average value for the concerning attribute. These metrics were the Number of Rules, Abstaining Rate, F1 score, Accuracy, Tentative Accuracy, and Execution Time. The descriptions of the metrics can be found in Section 3.7.1.

Please consider that there is a relationship between accuracy, tentative accuracy and abstaining rate. The accuracy counts abstained instances as misclassification, and tentative accuracy does not include abstained instances. Therefore, the higher the abstaining rate, the lower the accuracy and the higher the tentative accuracy.

4.6.2 Implemented Versions of Original PRISM using Different Types of Local and Global Discretisation Methods

Three different versions of original PRISM are implemented in this research for comparative purposes. All these versions are based on discretisation methods used to handle continuous attributes and varied only in the type of discretisation. G-Rules-IQR was compared against these variations of PRISM, which are briefly described as follows:

1. ***Prism-CutP:*** PRISM algorithm using a binary splitting and cut-point calculations (top-down local discretisation method). It is similar to the extended version of PRISM algorithm as introduced in [69, 71]. This implementation is referred to in this work as original PRISM algorithm.

2. ***Prism-ChiM:*** PRISM algorithm using ChiMerge (bottom-up global discretisation method), which is a well-known approach used to deal with continuous attributes in classification tasks [138].

3. ***Prism-CAIM:*** PRISM algorithm using CAIM (top-down global discretisation method), which does not require user defined parameters [139]. It is determined as the interdependency between the target class and the discretisation scheme of a continuous attribute.

4.6.3 Results and Interpretation

Tables 4.2 to 4.7 show the results of the experiments in terms of 6 evaluation metrics. In each table the '#' symbol indicates the index of the dataset in Table 4.1. *'T'* denotes that the transformation to normal distribution was switched on. The best result(s) in the tables for each dataset are highlighted in bold letters.

Table 4.2: G-Rules-IQR Experimental Results: Number of Rules

#	Prism			G-Prism				G-Rules IQR	
				DB		FB			
	CutP	ChiM	Caim		T		T		T
1	12	**8**	9	9	10	20	21	18	18
2	**17**	27	22	30	27	73	71	31	22
3	15	18	**11**	16	21	55	38	26	13
4	18	48	**7**	46	17	109	58	60	20
5	14	196	**8**	176	250	466	483	101	89
6	57	72	83	**44**	52	108	97	91	53
7	**51**	556	537	219	117	511	218	270	132
8	**110**	412	223	465	430	1236	1325	205	215
9	**24**	78	46	41	49	122	122	67	57
10	**17**	33	31	23	**17**	42	44	32	28
11	56	63	64	45	40	75	81	67	**30**
12	77	789	39	3928	2074	6292	7107	894	**31**
13	**25**	3929	129	3133	4563	7177	8281	3467	155
14	**74**	1827	1511	923	577	1576	1229	1643	171
15	868	3801	3901	843	**320**	2334	844	2600	875
16	33	41	37	23	**6**	48	9	49	11
17	30	32	31	30	30	30	30	**29**	**29**
18	**37**	3706	516	2602	4650	5603	7009	4585	4423

Table 4.3: G-Rules-IQR Experimental Results: Abstaining Rate

#	Prism			G-Prism				G-Rules IQR	
				DB		FB			
	CutP	ChiM	Caim		T		T		T
1	0.09	**0.00**	0.04	0.04	0.02	0.02	0.02	0.07	0.07
2	0.17	0.05	**0.03**	0.10	0.10	0.06	0.05	**0.03**	**0.03**
3	0.17	0.11	0.08	0.11	**0.04**	0.19	**0.04**	0.17	0.06
4	**0.00**	0.02	**0.00**	0.10	0.02	0.23	**0.00**	0.08	**0.00**
5	**0.00**	0.10	**0.00**	0.08	0.13	0.09	0.09	**0.00**	0.02
6	0.14	0.11	0.12	0.24	0.10	0.31	0.16	0.20	**0.08**
7	**0.07**	0.17	0.16	0.36	0.08	0.45	0.08	0.36	**0.07**
8	**0.01**	0.05	**0.01**	0.03	0.03	0.03	0.03	0.04	0.02
9	0.50	0.26	**0.03**	0.17	0.12	0.28	0.28	0.19	0.30
10	0.31	**0.06**	**0.06**	0.31	0.19	0.44	0.22	0.19	0.19
11	0.11	**0.08**	0.20	0.54	0.12	0.46	0.14	0.42	0.11
12	**0.00**	0.03	**0.00**	0.02	**0.00**	0.03	**0.00**	0.01	**0.00**
13	**0.00**	0.23	**0.00**	0.31	**0.00**	0.34	0.01	0.16	**0.00**
14	**0.00**	0.22	0.12	0.40	0.01	0.35	0.02	0.41	0.01
15	0.38	0.13	0.14	0.16	**0.01**	0.16	0.03	0.15	0.04
16	0.02	0.01	**0.00**	0.03	**0.00**	0.02	0.02	0.02	**0.00**
17	**0.04**	**0.04**	0.11	**0.04**	**0.04**	**0.04**	**0.04**	0.11	0.11
18	**0.00**	0.16	0.01	0.35	0.15	0.33	0.12	0.23	0.19

Tabel 4.2 demonstrates the results of the number of rules generated. A small number of rules is desired. This is the only metric where the basic Prism (particularly Prism-CutP) outperforms G-Rules-IQR classifier and G-Prism classifiers. Nevertheless, comparing with G-Prism-FB and G-Prism-DB approaches, G-Rules-IQR is outperformed on the transformed data on all the datasets and in some cases it produces fewer rules than original Prism.

Table 4.3 shows the abstaining rate results. As previously explained in Section 3.7.1, the abstaining rate for rule-based predictions is the percentage of data instances remaining unclassified due to no matching rules being available. This may be a desirable feature in critical applications where a false classification is very costly, such as in health, safety and finance. However, the lower the abstaining rate the better for most applications. In general, the results show that there is no clear winner in terms of abstaining rate. However, Prism-CutP and Prism-Caim abstain less than their competitors, as it can be seen in Table 4.3 and Figure 4.5.

Table 4.4: G-Rules-IQR Experimental Results: F1 score

#	Prism			G-Prism				G-Rules IQR	
				DB		FB			
	CutP	ChiM	Caim		T		T		T
1	0.93	0.91	0.95	0.93	0.91	0.93	0.93	**0.96**	**0.96**
2	0.96	0.97	0.95	0.93	**1.00**	0.89	**1.00**	0.94	**1.00**
3	**0.98**	0.93	0.92	0.96	**0.98**	**0.98**	**0.98**	0.89	**0.98**
4	0.87	0.87	0.87	0.89	**1.00**	0.90	**1.00**	0.89	0.98
5	0.80	**0.99**	0.94	0.96	0.97	0.96	0.97	0.98	**0.99**
6	0.77	0.61	0.71	0.72	**0.80**	0.72	0.61	0.62	0.79
7	0.33	0.53	0.55	0.49	0.75	0.49	0.81	0.54	**0.86**
8	0.64	0.74	0.78	0.80	0.84	0.82	0.85	0.89	**0.93**
9	0.82	0.91	0.87	0.92	0.86	0.84	0.84	0.94	**0.96**
10	0.81	0.73	0.83	**0.93**	0.93	0.79	0.77	0.80	0.81
11	0.64	0.73	0.84	0.67	**0.97**	0.44	0.90	0.61	0.86
12	0.96	0.99	0.99	0.99	**1.00**	0.99	**1.00**	0.99	**1.00**
13	0.80	0.98	0.85	0.88	0.95	0.87	0.95	0.91	**1.00**
14	0.29	0.49	0.35	0.50	**0.95**	0.50	0.79	0.55	0.79
15	0.90	0.82	0.83	0.87	**0.99**	0.88	**0.99**	0.88	**0.99**
16	0.97	0.97	0.97	0.97	**1.00**	0.98	**1.00**	0.98	**1.00**
17	0.38	0.53	**0.69**	0.49	0.49	0.49	0.49	0.52	0.52
18	0.71	0.83	0.76	0.79	0.79	0.77	0.78	**0.87**	0.86

Table 4.4 details the results of the F1 score for each of the classifiers in the experiments. As can be seen in Equation 4.2, F1 score is the harmonic mean of precision and recall. In multi-class classification problems, precision and recall are computed by taking an average of these metrics' values for each class.

$$\text{F1 score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.2}$$

The results show that G-Rules-IQR classifier with transformation aspect achieved

the best score on 11 out of 18 datasets. For most of the remaining 7 datasets where G-Rules-IQR-(T) did achieve the best F1 score, it was still close to the best performing F1 score, in particular for 3 datasets it was at most only 3% lower than the best F1 score.

Table 4.5: G-Rules-IQR Experimental Results: Accuracy

#	Prism			G-Prism				G-Rules IQR	
				DB		FB			
	CutP	ChiM	Caim		T		T		T
1	0.87	**0.91**	**0.91**	**0.91**	**0.91**	**0.91**	**0.91**	**0.91**	**0.91**
2	0.92	0.95	0.94	0.86	0.90	0.87	0.95	0.87	**0.97**
3	0.89	0.83	0.85	0.89	**0.96**	0.79	0.94	0.85	0.94
4	0.76	0.77	0.77	0.77	0.98	0.77	**1.00**	0.77	0.97
5	0.72	0.93	0.94	0.92	0.92	0.90	0.93	**0.98**	**0.98**
6	0.77	0.75	0.75	0.72	0.85	0.65	0.77	0.73	**0.91**
7	0.37	0.51	0.51	0.44	0.87	0.46	**0.89**	0.49	**0.89**
8	0.95	0.95	0.96	0.95	0.97	0.95	0.97	0.96	**0.98**
9	0.61	0.74	**0.83**	0.78	0.76	0.66	0.66	0.82	0.72
10	0.59	0.72	**0.81**	0.66	0.78	0.50	0.69	0.66	0.66
11	0.60	0.77	0.75	0.55	**0.86**	0.51	0.83	0.58	**0.86**
12	0.92	0.97	0.98	0.97	**1.00**	0.96	0.99	0.98	**1.00**
13	0.67	0.86	0.80	0.74	0.94	0.72	0.93	0.80	**1.00**
14	0.49	0.63	0.58	0.56	0.98	0.59	0.97	0.60	**0.99**
15	0.57	0.72	0.71	0.73	**0.98**	0.75	0.96	0.75	0.96
16	0.96	0.95	0.96	0.94	**1.00**	0.96	1.00	0.96	**1.00**
17	0.59	**0.74**	**0.74**	0.63	0.63	0.63	0.63	0.67	0.67
18	0.56	0.75	0.70	0.67	0.71	0.66	0.72	**0.77**	**0.77**

Table 4.5 illustrates the results of the accuracy for each classifier implemented in the experiments. Please note that instances that have not covered in the rule set are classified using the majority class strategy and counted in this metric. G-Rules-IQR (T) accomplished the best accuracy in 12 out of 18 datasets. On 3 out of the remaining 6 cases, G-Rules-IQR was not the best but was still very close within 3% of the best accuracy. On the other remaining 3 datasets (9 , 10 and 17), G-Rules-IQR's accuracy was much lower than the other evaluated classifiers. However, these datasets also cause a relatively high abstaining rate, and therefore they had been classified using the majority class method.

Table 4.6 lists the results of the tentative accuracy. G-Rules-IQR classifier with transformation achieved the highest tentative accuracy on 13 out of 18 datasets and on 3 out of the remaining 5 datasets its accuracy was within 3% of the best tentative accuracy.

Table 4.7 demonstrates the results of the execution times. These also include the time needed approximating normal for G-Prism classifiers and G-Rules-IQR. As can be seen in the table, G-Rules-IQR with transformation achieved shortest execution times on 15 out of 18 datasets. Thus, it outperforms the remaining G-Prism classifiers and all three versions of basic Prism algorithm.

Table 4.6: G-Rules-IQR Experimental Results: Tentative Accuracy

#	Prism			G-Prism				G-Rules IQR	
				DB		FB			
	CutP	ChiM	Caim		T		T		T
1	0.93	0.91	**0.95**	0.93	0.91	0.93	0.93	**0.95**	**0.95**
2	0.96	0.97	0.95	0.93	**1.00**	0.93	**1.00**	0.89	**1.00**
3	**0.98**	0.94	0.92	0.96	**0.98**	0.88	**0.98**	**0.98**	**0.98**
4	0.76	0.78	0.77	0.81	**1.00**	0.82	**1.00**	0.80	0.97
5	0.72	**0.99**	0.94	0.96	0.97	0.95	0.97	0.98	**0.99**
6	0.86	0.81	0.83	0.86	0.91	0.86	0.86	0.84	**0.94**
7	0.40	0.54	0.57	0.54	0.95	0.58	0.96	0.59	**0.97**
8	0.95	0.97	0.97	0.96	0.98	0.96	0.97	0.98	**0.99**
9	0.83	0.91	0.86	0.91	0.85	0.83	0.83	0.94	**0.95**
10	0.86	0.77	0.87	**0.95**	0.92	0.89	0.84	0.81	0.81
11	0.64	0.82	0.87	0.70	**0.98**	0.54	0.96	0.71	0.97
12	0.92	0.99	0.98	0.98	**1.00**	0.98	0.99	0.98	**1.00**
13	0.67	0.97	0.80	0.82	0.94	0.80	0.93	0.87	**1.00**
14	0.49	0.66	0.61	0.63	0.99	0.66	0.98	0.69	**1.00**
15	0.90	0.82	0.82	0.87	**0.99**	0.88	**0.99**	0.88	**0.99**
16	0.97	0.96	0.96	0.97	**1.00**	0.97	**1.00**	0.98	**1.00**
17	0.62	0.73	**0.75**	0.65	0.65	0.65	0.65	0.67	0.67
18	0.56	0.79	0.71	0.74	0.73	0.72	0.72	**0.84**	0.83

Table 4.7: G-Rules-IQR Experimental Results: Execution Time (in seconds)

#	Prism			G-Prism				G-Rules IQR	
				DB		FB			
	CutP	ChiM	Caim		T		T		T
1	2.66	2.18	2.39	1.92	**1.65**	2.98	2.61	1.95	**1.65**
2	3.86	12.11	8.10	5.97	5.52	4.46	4.17	3.07	**2.63**
3	4.93	11.27	8.52	8.09	5.86	4.73	4.11	3.93	**2.75**
4	6.10	15.44	7.86	12.22	**5.33**	11.20	6.90	8.33	**5.33**
5	15.07	224.80	25.31	32.70	31.64	38.50	38.78	15.64	**13.48**
6	10.76	18.14	22.62	9.69	6.58	8.37	6.54	7.89	**5.23**
7	40.37	262.30	394.20	98.40	26.08	100.20	30.32	76.20	**23.60**
8	1068.00	546.00	407.40	295.80	256.80	440.40	417.00	115.20	**87.00**
9	12.29	21.77	12.49	7.78	7.46	6.51	6.25	4.98	**4.25**
10	24.37	19.60	13.77	6.78	4.55	3.11	3.12	3.50	**2.88**
11	39.87	18.62	27.06	9.26	5.41	6.44	4.63	7.37	**3.18**
12	352.80	65520.00	2492.40	6600.00	3066.00	5652.00	7200.00	1055.00	**165.60**
13	568.80	234144.00	3321.00	6500.00	5184.00	6768.00	6840.00	6372.00	**238.80**
14	198.00	1622.00	2428.20	509.40	248.40	459.60	330.60	1079.00	**86.40**
15	1720.00	10800.00	11844.00	6000.00	**754.80**	2583.00	939.00	3182.00	1049.00
16	11.94	12.82	13.72	11.97	5.48	9.04	5.76	8.90	**5.03**
17	3.66	28.22	5.88	4.28	**3.20**	3.64	3.50	4.36	3.53
18	**262.20**	8028.00	1752.00	6540.00	5472.00	4464.00	5508.00	10224.00	10296.00

4.6.4 Runtime Complexity (Big O Notation)

Overall, the time complexity of G-Rules-IQR is expected to be similar to that of G-Prism and basic Prism algorithms in terms of the number of instances N and number of attributes M. However, G-Rules-IQR is faster than the others. The authors of [110] estimated the worst case time complexity of a PRISM classifier to be approximately $O(N^2 \cdot M)$. In the worst case, each rule covers exactly one data instance and each rule has two rule-terms per attribute. This is a very unlikely case and time complexity is strongly dependent on the pattern in the data that can be expressed in the form of rules. The worst case of G-Rules-IQR and GPrism

classifiers would induce only 1 rule-term per attribute, and thus already divides the worst case complexity by 2. Also, G-Rules-IQR is expected to be faster than G-Prism-DB and Prism because of the number of calculations required to induce a rule-term. In the worst case scenario, Prism and G-Prism have to evaluate either several cut-point calculations or rule-term boundaries, whereas G-Rules-IQR only has to calculate the quartiles. Also, G-Rules-IQR with transformation has a lower runtime than G-Rules-IQR even though there is an additional operation. However, this is likely because, in most cases, G-Rules-IQR with transformation produces fewer rules than G-Rules-IQR without transformation.

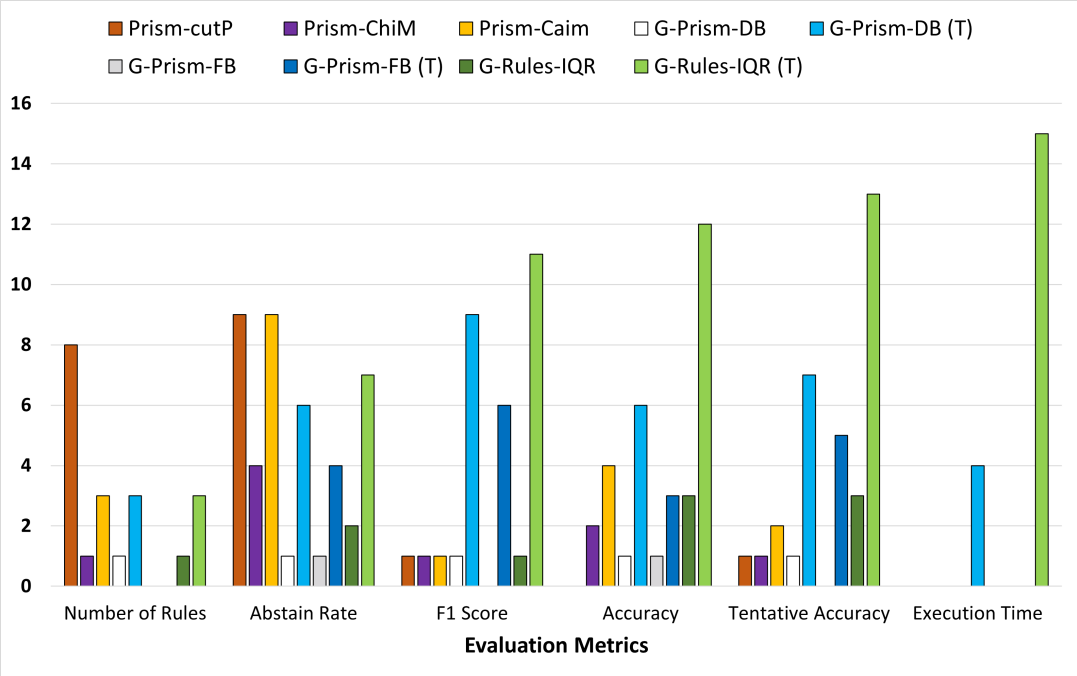


Figure 4.5: Summary of results: how often a particular algorithm achieved the best results comparing with its competitors

4.7 Summary

This chapter proposed the rule-based G-Rules-IQR algorithm for continuous attributes. The chapter began with investigating the limitations of G-Prism-FB and G-Prism-DB algorithms, which were developed in the previous chapter and published in [19] and in [20] respectively. These limitations are: (1) the more accurate G-Prism-DB requires user defined rule-term boundary threshold, (2) both approaches assume normally distributed continuous attributes, (3) both algorithms have a higher abstaining rate than the original Prism classifier and (4) the more accurate G-Prism-DB was estimated to have a longer execution time than G-Prism-FB and the basic Prism. With respect to underfitting rule-term boundaries, Section

4.5.2 proposed a method that optimises these boundaries by replacing the user defined threshold with IQR based rule-term boundaries. In terms of the assumptions of normally distributed data, Section 4.5.1 introduced an approximation towards normally distributed data.

As shown in Figure 4.4, G-Rules-IQR algorithm integrates the aforementioned methods in addition to the GPDD-based approach, which was developed in Chapter 3 and a testing for normality approach which was introduced in Section 4.4. The test was carried out prior to the application of G-Rules-IQR. In Section 4.6, G-Rules-IQR algorithm was evaluated empirically and comparatively with G-Prism-FB, G-Prism-DB and the original Prism with various well known discretisation methods (binary splitting, ChiMerge and Caim). For comparative purposes, the implementations of G-Rules-IQR and G-Prism algorithms allowed to switch off the transformation to approximate normal distribution. No transformation has been implemented for the three variations of original Prism, as they do not assume normally distributed continuous attributes, and they convert them into categorical by discretising their values.

Overall, as demonstrated in Figure 4.5, G-Rules-IQR with transformation outperformed its competitors with respect to F1 score, accuracy, tentative accuracy and execution time. With regard to limitation (1), G-Rules-IQR achieves shorter execution times than its competitors, with respect to limitation (2) G-Prism achieves a competitive (similar) abstaining rate as its competitors, with respect to limitation (3) G-Rules-IQR does not require user input for rule-term boundary thresholds and with respect to limitation (4) the normal distribution approximation made G-Rules-IQR the best performing Prism based classifier in this chapter. Thus, G-Rules-IQR will be used for the development of the ensemble learner described in Chapter 5.

Chapter 5

ReG-Rules: an Explainable Rule-based Ensemble Learner

This chapter introduces a new framework of an explainable rule-based ensemble learner, called ReG-Rules (Ranked ensemble G-Rules), which consists of 5 components (stages) with several operations. The chapter also, illustrates the three new methods incorporated in ReG-Rules's construction. Then, several experimental studies are presented to evaluate empirically and qualitatively ReG-Rules learner and its integrated methods. The ensemble ReG-Rules learner is one of the contributions of this project, and it is published in [22].

5.1 Introduction

In Chapter 4, a new rule-based predictive algorithm (G-Rules-IQR) has been developed. The algorithm integrates testing for normality for each attribute in the dataset, as previously shown in Figure 4.4. Depending on the results of this normality testing, G-Rules-IQR algorithm applies an approximate normal transformation of the attribute's values with respect to a target class. The algorithm was empirically evaluated in Section 4.6, comparing its performance with five different Prism based approaches. The results revealed that G-Rules-IQR classifier with transformation outperformed its competitors in most cases. Therefore, the choice was made to use the G-Rules-IQR algorithm as the base inducer of the ensemble classifier that is described and evaluated in this chapter with the goal of maximising the overall accuracy as well as maintaining a high level of explainability in terms of rule examinations needed for tracing individual predictions.

Therefore, objective 3 (restated below) is fully met in this chapter:

> *"To improve the quality of rule sets by developing a rule merging technique for predictive rules and minimising loss of accuracy.'*

Also, the chapter is employing steps to meet the research objective number 4 restated below:

> *"To develop an expressive ensemble learner footed upon the base classifier developed on objective 2 and the Rule Merging technique in objective 3."*

This chapter begins with brief descriptions of some limitations related to standalone learning systems, which emphasise the strength of utilising ensemble learning to address such issues. Next, Section 5.3 introduces the framework of the ensemble learning system termed 'ReG-Rules' which consists of 5 components (stages) with several operations. Each stage is illustrated in a separate section in this chapter.

Section 5.3.1 provides the description of component (1) Diversity Generation. As Figure 5.1 shows, it is based on the method of manipulating a dataset by using different samples of the training data to train individual classifiers. Specifically, to obtain classifiers diversity, ReG-Rules classifier integrates bagging method [9] in its construction stage.

Section 5.3.2 illustrates component (2) Base Classifiers Induction (see Figure 5.1), where multiple learners are generated independently using the aforementioned inducer (G-Rules-IQR algorithm). These base learners are weighted in this stage using a 'performance weighting method', which consists of a combination of measurements that can be obtained during a validation phase.

The component (3) Models Selection (see Figure 5.1) is thoroughly described in Section 5.3.3 where the number of base classifiers that will be participating in the final ensemble predictions is reduced based on the theorem of 'many could be better than all' [124] by ranking all the classifiers first according to a certain criterion using a proposed ranking-based method. Then ReG-Rules selects all the models that are above a particular threshold. This ranking approach is evaluated in Section 5.4.4.

Regarding component (4) Rules Improvement (see Figure 5.1), this stage involves improving the quality of the rules locally and independently for each top-ranked selected models. Section 5.3.4 explains this ReG-Rules component and its proposed integrated Rule Merging method is explained using three exemplary scenarios.

In terms of component (5) Combination and Prediction (see Figure 5.1), Section 5.3.5 outlines the combination method, which is introduced in this chapter and based on a weighted voting strategy. For this, ReG-Rules builds a committee of rules for each new unlabelled instance to decide its final predicted class label.

Finally, Section 5.4 demonstrates the experiments to evaluate the approaches developed in this chapter. The general performance of the ensemble ReG-Rules is evaluated empirically in Section 5.4.3 and compared with the stand-alone G-Rules-IQR classifier. The new proposed method that has been integrated in ReG-Rules and is used for ranking and selecting models is empirically evaluated in Section 5.4.4. While the other new proposed methods that have been developed to improve the quality of the rules are evaluated in Section 5.4.5.

5.2 Issues with Stand-alone Classification Systems

Despite that G-Rules-IQR algorithm shows high performance compared with its competitors, in general, stand-alone classifiers are not stable, especially when they are applied on unbalanced or noisy datasets. Also, they are sensitive to sampling techniques and consequently the level of predictive accuracy varies between different samples [155]. Accordingly, as mentioned in Section 2.5.1, learning algorithms that produce only a single classification model suffer from the following three essential drawbacks: (1) the statistical problem; (2) the computational problem; (3) and the representational problem.

The statistical issue occurs when a learning algorithm is dealing with a too small or a too large amount of training instances. In such cases, different classification models with similar predictive accuracy rates might be generated and hence selecting one of them would be a difficult task as the risk of choosing an overfitted model is rather high [2]. The computational issue is also related to the size of the dataset. In real life, there might be considerable dependencies between different features, especially in datasets with high dimensionality. Therefore, like with the statistical problem, finding the best model in a feasible execution time might be very challenging. The representational issue occurs when there is no ideal classifier to be selected within the space of all possible classification models.

Generally, a learning model that suffers from statistical or computational issues is described as having a high variance while a model that experiences representational problems is said to have a high bias. According to [155], constructing an ensemble classification model by combining the predictions from several base classifiers can be an effective method to overcome the aforementioned problems, as the main strength of ensemble learning is the ability to handle bias and variance in the data effectively.

5.3 Framework for the Ensemble Learning System: ReG-Rules

This section proposes a new rule-based ensemble classification system named: **R**anked **e**nsemble **G-Rules**-IQR (ReG-Rules). The aim of this system is to improve the predictive performance of explainable rule-based learners while presenting the human analyst with a readable model for predictions. Algorithm 6 details the pseudocode for ReG-Rules classifier and Figure 5.1 demonstrates the general framework of the system, which consists of five stages with several operations: (1) Diversity Generation, (2) Base Classifier Inductions, (3) Models Selections, (4) Rule Merging Technique, (5) Combination and Prediction. These stages will be explained in the following sections, referring to the lines of code in Algorithm 6.

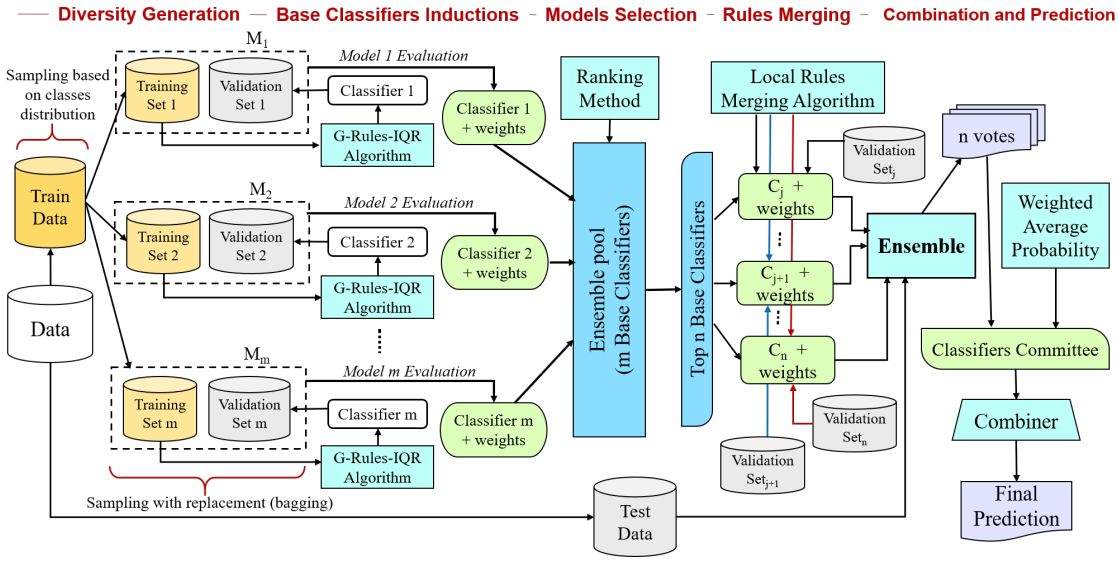


Figure 5.1: The General framework of the ensemble rule-based classifier ReG-Rules

5.3.1 Stage 1: Ensemble Diversity Generation

The performance of an ensemble classification model is heavily reliant on the level of diversity among the group of classifiers, which comprises the ensemble. Combining several models with identical or even similar outputs to produce a do-nothing ensemble model. Therefore, the generation of a set of base classifiers should be as diverse as possible to assure producing uncorrelated errors and then obtain a more accurate ensemble. However, there is no theoretical explanation of how and why inducing diverse individual base models contribute to overall ensemble classifier [24, 73, 92].

Algorithm 6: Ensemble Rule-based Classifier: ReG-Rules

Notations: S: training dataset, M: original ensemble size, n: number of top-ranked base classifiers defined by user/default ,
V: validation dataset, R: rule set, BC: base classifier,
E_{pool}: ensemble pool, w: weight, $topBC$: top-ranked base classifier

1 Initialise the ensemble model (ReG-Rules)
2 **for** $i = 1 \rightarrow M$ **do**
3 $s_i \leftarrow$ Random sample with replacement using Bagging method
4 $v_i \leftarrow$ out-of-bag set
5 Generate a base classifier BC_i by **applying Algorithm 5 (G-Rules-IQR)** on s_i dataset and learn a rule set $\rightarrow R_i$
6 Evaluate BC_i performance by applying R_i on v_i dataset
7 Calculate a weight for each rule induced in previous line
8 Send BC_i including its rule set weights to the ensemble pool E_{pool}
9 **end**
10 Rank all the base classifiers BC collected in E_{pool} according to the criteria described in Section 5.3.3
11 Eliminate weak BC by selecting the n top models $(topBC)$ ranked in the previous step according to the following if statement:
12 **if** *models selection type = defualt* **then**
13 $n \leftarrow 20\%\ M$ models
14 **else**
15 $n \leftarrow$ selected models size defined by user
16 Select the top n BC models in line 10
17 Assign all the n classifiers $(topBC)$ selected in line 11 to the ensemble model (ReG-Rules)
18 **for** $j = 1 \rightarrow n$ **do**
19 $w_1 \leftarrow$ R_j weight computed previously in line 6
20 **Apply Algorithm 8 (Local Rule Merging)** on current $topBc_j$ and update its rule set R_j
21 Re evaluate R_j on the same validation dataset used for weighting the rules in line 6
22 $w_2 \leftarrow$ Calculate the merged rules R_j weight returned from the previous line
23 **if** $w_2 > w_1$ **then**
24 Replace rule set of the current $topBC_j$ by the new merged rules R_j
25 **end**
26 Sort the rule set R_j according to their correctly used times
27 **end**
28 **return** *ReG-Rules Classifier*

There is a number of popular methods that can be used for creating diversity in ensembles, such as manipulating the inducer, manipulating the training samples, manipulating input features, and manipulating output representation. The reader is referred to Chapter 2 (Section 2.6.2) for brief descriptions about these methods.

ReG-Rules is a homogeneous ensemble in which all its base classifiers are generated using the same base algorithm (G-Rules-IQR). Also, the system adopts the parallel learning approach, i.e. the induction of each base learner is independent and can be built in parallel to other models without cooperation in the training phase. The reason for adopting the parallel ensemble methodology is that they are perfectly suitable to parallel computing in which the speed and memory constraints can be addressed by distributed environments as previously discussed in Section 2.6.1. However, parallelising ReG-Rules is outside the scope of this thesis.

Accordingly, the method utilised in this thesis to train diverse classifiers in ReG-Rules ensemble system is: 'using different training datasets to train individual classifiers'. In this approach all the subsets are drawn from a single data source, but they can just as well be entirely different samples taken from different sources capturing different features of data if a suitable level of randomness is used [92]. Specifically, as shown in Figure 5.1 (diversity generation part), ReG-Rules system uses two types of training sampling in order to maximise the level of base classifiers' diversity:

1. Sample a dataset randomly without replacement into train and test datasets.

2. Bagging, which is a well-known sampling with replacement method.

The reader is referred to Section 2.6 for further details about bagging. Regarding the first sampling method, note that the test data is used only once to assess the general performance of the whole ensemble model. With respect to the second sampling method (bagging), it is used in to produce multiple data samples. The size of each sample is equal to the size of the training dataset, and then some instances may appear more than once and some may not appear at all. Thus, statistically, a sample created using the bagging method is likely to contain 63.2% of the original training dataset which can be used to construct a base classifier. The remaining 36.8% of the original instances that are not picked in the training phase and called out-of-bag instances which can be used as a validation dataset to independently assess the performance of the base classifier [9]. Bagging method is integrated in ReG-Rules system as shown in lines (3 and 4) of Algorithm 6.

5.3.2 Stage 2: Base Classifiers Inductions

This stage includes a significant factor that controls the induction of any ensemble model, the number of base classifiers that should be generated (ensemble size). This is simply a predefined parameter that can be set by the user and is represented by the symbol 'M' in Figure 5.1 (see Algorithm 6, line 2). However, determining a proper ensemble size requires balancing accuracy and efficiency. Thus, it can not be too small to ensure a high level of base classifier diversity, and also it can not be too large to avoid the high cost in terms of computations and memory resources [24, 121]. There is no ideal ensemble size, and the impact of this user-defined number on the ensemble performance makes its determination even more difficult [114]. However, a major experimental study conducted in [121] suggested constructing between 64 and 128 base classifiers to maintain a balance between computational cost and accuracy in most cases. The same study concluded that there is no significant performance gain if a larger number of base

models is generated. Accordingly, this thesis uses a 100 G-Rules-IQR base classifiers, which is within this range, as a default number in the experiments presented in this thesis.

The inducer, G-Rules-IQR (Algorithm 5) is invoked in line 5 of Algorithm 6 to independently train M base classifiers on M random training samples generated by bagging method. Then, for the purpose of finding the best learners that can be chosen within a smaller ensemble, ReG-Rules algorithm (line 6) assesses the performance of each individual base classifier using different validation data subsets, i.e. out-of-bag instances produced by the bagging method. This independent evaluation process during the induction stage of base classifiers is called a *'classifier's performance weighting'*, which will be described in the next section.

Base Classifier's Performance Weighting

The predictive accuracy is considered to be the key evaluation criterion in most classification methods. However, in many real-life datasets, especially in imbalanced domains, accuracy alone might be an insufficient metric to evaluate the performance of a classifier [113]. Hence, a more suitable metric or a set of metrics must be carefully selected to provide the more reliable weighting of the classifier's performance and avoid the bias of the training model (overfitting). This combination of measurements can be obtained during the validation phase in ReG-Rules ensemble model. In other words, given M base classifiers are induced in the training phase, their evaluation criteria are organised as an M-dimensional vector, which consists of the following:

1. Rule set size: the number of rules induced for each base classifier using the inducer (G-Rules-IQR) algorithm.

2. Average of a rule length: this is the average number of terms per rule for each base classifier.

3. CUR: stands for 'Correctly Used Rules', which is the number of times a rule was used during the validation phase and predicted the correct class label. This is a *track record prediction* for each rule separately in each base classifier.

4. Abstaining rate: the proportion of instances a base classifier abstains from classification, i.e. the proportion of validation instances not covered in the rule set for each base classifier.

5. Accuracy: the ratio of validation instances that have been correctly classified either using the rule set or majority class strategy for each base classifier.

6. Tentative accuracy: the ratio of validation data that have been correctly classified based only on the number of instances that have been assigned a classification.

The final step of this stage of ReG-Rules classifier construction is represented by the terms *'ensemble pool'* in Figure 5.1 and E_{pool} in Algorithm 6 (line 8).

5.3.3 Stage 3: Models Selection

As previously discussed in Chapter 2 (Section 2.6.4), the number of component classifiers that should be included in the final model is a crucial factor for building an effective and accurate ensemble [24,114]. A large ensemble can access and examine different feature sub-spaces, which might increase its general classification accuracy. However, it might be computationally inefficient and more likely more difficult to understand for human. This potential problem could be addressed or mitigated by reducing the number of base classifiers that can participate in the final ensemble predictions according to the theorem of *'many could be better than all'* which was introduced in [124]. Nevertheless, determining the exact amount of reduction in the ensemble size without causing considerable loss in its accuracy is a difficult choice. Therefore, inspired by the aforementioned theorem many ensemble models selection approaches have been developed in the literature [125]. Amongst these were the two widely used methods: 'ranking-based' and 'search-based' that have been reviewed in Chapter 2 (Section 2.6.4).

Generally speaking, search-based method is suitable for sequential ensemble systems as additional members are gradually added to the ensemble subset since its performance is sequentially increasing. Alternatively, the main concept of the other method, ranking-based, is to separately rank each base classifier according to a certain criterion and then chose the models that are above a particular threshold [73]. This makes ranking-based models selection method more suitable for the parallel ensemble systems. Therefore, a newly developed ranking-based approach, termed *'Ranking-based CUR'* is proposed in this chapter, which can be integrated in ReG-Rules algorithm and used to rank its base classifier. This is further illustrated below.

Ranking-based CUR Approach.

ReG-Rules system makes use of the combination of metrics that are obtained during the base classifiers' performance weighting stage, which was explained in Section 5.3.2. Specifically, three of these metrics, namely (1) tentative accuracy, (2) CUR and (3) abstaining rate, are used as ensemble selection criteria by ranking all

the base classifiers accordingly. First, all the models are ranked in descending order according to their tentative accuracies and in case of tie-breaks the descending order will be based on their average CUR values. Next, if more tie-breaks occur, the ranking will be continued based on the ascending order of their abstaining rates. Then, as highlighted in Algorithm 6 (lines 10 and 11), when the ranking process is completed the weak base classifiers will be eliminated after selecting the top-ranked models ($topBC$) according to a predefined size (n). This size is determined in ReG-Rules algorithm using two types of threshold: (1) default or (2) user defined. There is no ideal ensemble size to be specified [114] but in this project, the default threshold is the top 20% of the ranked models, and it was set to ensure that only the strong base classifiers are selected. Thus, for example, from the 100 base classifiers induced in the experiments presented in this chapter, only the top 20 ranked base classifiers are chosen to design the final ReG-Rules ensemble system and the remaining 80 models are discarded. Despite this big reduction in the ensemble size, the top 20 models were sufficient according to 'many could be better than all' theorem [124] and this default threshold worked well in most cases investigated in this chapter.

5.3.4 Stage 4: Rules Improvement using Local Rules Merging Algorithm

As mentioned in Chapter 2 (Section 2.4.5), the main difference between the rules generated by a tree (using divide and conquer strategy) and the rules generated by a rule learning algorithm (using separate and conquer strategy), is that the induction of the former rule set follows the 'theory of non-overlapping rules'. Imposing such a restriction on rules mostly results to inducing a classification rule set that contains redundancy (see Figure 2.6 and Rule set 2.2).

Separate and conquer rule induction algorithms such as G-Rules-IQR and Prism family relax this constraint by allowing for possibly overlapping rules, which may often result in smaller rule sets that are less susceptible to the redundancy problem during the training phase [3]. However, overlapping rules are generally unnecessary as they need to be tested at the prediction stage, thus incur unnecessary computational cost of classification. Also, multiple overlapping rules may cause conflicting predictions for the same instance [3].[1]

Therefore, the ensemble ReG-Rules learner integrates a new method termed local Rule Merging (RM) with the aim to address locally and independently the is-

[1]The strategies that are often used to solve the classification conflicts in most separate and conquer rule induction algorithms, are previously described in Section 3.3.2.

sue of overlapping rules for each selected base classifier, i.e. improving the quality of induced rule sets. As can be seen in Algorithm 6, the process of improvement starts in line 20 by applying the RM algorithm for each rule set. Then a new improved rule set denoted by R_j is evaluated on the same validation dataset that have been used for weighting the original rule set. Thus, a new weight (w_2) will be assigned to the newly merged rule set as shown in line 22 and by comparing it with the previous weight (w_1) that has been computed in line 7, ReG-Rules decides whether to keep the current original rule set or replace it with the new improved (merged) one. In both cases, the resulting rule set R_j of the current base classifier ($topBC_j$) will be sorted according to their performance during validation phase, as shown in line 26.

Integrated Local Rule Merging (RM) Algorithm

The local Rule Merging technique represents a post-processing of the induced rules, and it is applied on the rules of each target class in turn. The pseudocode of this technique is presented in Algorithm 8. The process begins with filtering rules according to their target class and the attributes contained in their rule-terms. Hence, some rules within the same target class will either be discarded or merged with other rules according to their similarities (overlap of features' ranges). This is conditioned by the decision returned from Algorithm 7 (Overlap Checking), which is invoked by the RM algorithm in line 7 to carry out the examination.

Algorithm 7: Overlap Checking

```
1  Input: Rule1 (current rule)
2         Rule2 (another rule)

3  if ( class label in Rule1 = class label in Rule2) and
4     ( all attributes α in Rule1 = all attributes α in Rule2) then
5     foreach attribute α ∈ Rule1 and Rule2 do
6        switch the type of α do
7           case Continuous do
8              if (lower bound of one rule includes the lower bound of the other and
9              upper bound of one rule includes the upper bound of the other) then
10                OverlapExist ← True
11             else
12                OverlapExist ← False
13          case Categorical do
14             if ( discrete value in Rule1 = discrete value in Rule2) then
15                OverlapExist ← True
16             else
17                OverlapExist ← False
18          end
19       end
20       if (OverlapExist = False ) then
21          Exit the loop in line 5
22       end
23    end
24 else
25    OverlapExist ← False
26 end
27 return OverlapExist
```

Overlap checking: both rules under examination must refer to the same class label and must contain the same attribute names in order to start the process of checking. In case of continuous terms, the lower bound in one rule should include the lower bound of the other rule. The same condition is applied to the upper bounds of the same rules. Whereas, in categorical terms, the discrete values of one rule should be equal to the other rule in order to be considered as overlapped rules. In case of having more than one term in each rule, all the corresponding terms should have passed the overlap checking together. The process will be terminated once any term fails the overlap checking. By the end of Algorithm 7, a decision (true/false) about the current rules' examination is retuned to the RM algorithm.

Next, if the overlap exist the merging is performed and results in more concise and smaller base classifier rule sets, which are thus expected to be more easily read and understood by human analysts. The following paragraphs describe the RM approach using three exemplary scenarios.

Algorithm 8: Local Rule Merging (RM) Algorithm

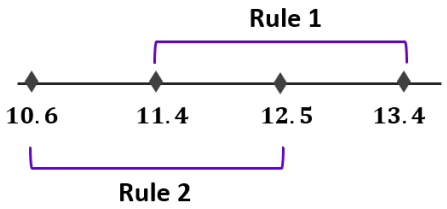
Notations: R: rule set,
$checkedRules$: rules that have been merged or checked,
$OtherR_j$: the remaining rules , α: attribute

1 $checkedRules \rightarrow$ empty
2 **for** $i = 1 \rightarrow \mathbb{R}$ **do**
3 $checkedRules \leftarrow ckeckedRules + R_i$;
4 $Other\mathbb{R} \leftarrow \mathbb{R}\ [-checkedRules]$;
5 $j = 1$;
6 **repeat**
7 $OverlapExist \leftarrow$ **Apply Algorithm 7 (Overlaps Checking)** on R_i and $OtherR_j$
8 **if** *($overlapExist$ = True)* **then**
9 Compute new upper and lower bounds for each rule-terms ;
10 Create merged rule in a form of $(x < \alpha \leq y)$ or $(\alpha = x)$;
11 Replace R_i in $\mathbb{R}$ rule set by a new merged rule created in line 10
12 **end**
13 $j \leftarrow j + 1$;
14 **until** *No more rules in $Other\mathbb{R}$ set*;
15 **end**
16 **return** *new rules list* $\mathbb{R}$

Figure 5.2 demonstrates the main idea of the Rule Merging algorithm using two basic examples where, in each example, two different rules sharing the same attribute and the same class label. In Figure (5.2a) there is an overlap between the two rules and therefore it can be merged into the single rule shown below in Rules 5.1.

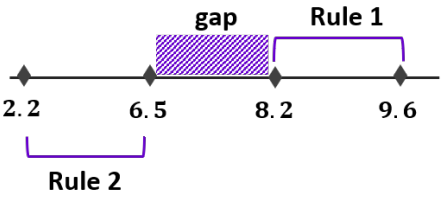
$$\textbf{Merged Rule: if (10.6 < } \alpha_1 \leq \textbf{13.4) Then low} \tag{5.1}$$

Rules 5.1: A rule produced after applying the local RM approach to the example of rules presented in Figure 5.2a.



Rule 1: if (11.4 < $\alpha_1 \leq$ 13.4) Then low
Rule 2: if (10.6 < $\alpha_1 \leq$ 12.5) Then low

(a) **overlapping rules**



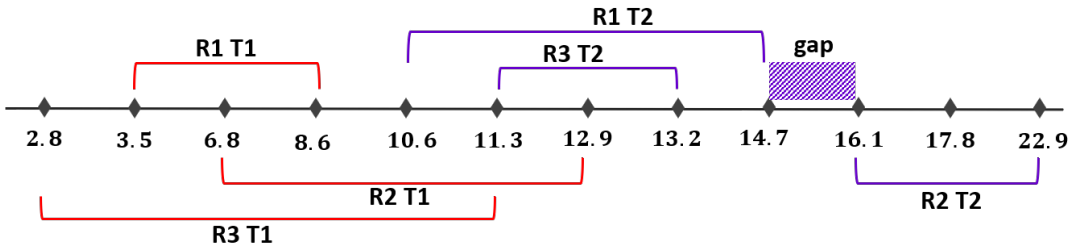
Rule 1: if (8.2 < $\alpha_1 \leq$ 9.6) Then high
Rule 2: if (2.2 < $\alpha_1 \leq$ 6.5) Then high

(b) **non overlapping rules**

Figure 5.2: Rule sets with single term each rule sharing similar features and classes.
In example **(a)** there is an overlap between rules and in example **(b)** the rules do not overlapped.

In case of Figure (5.2b), there is a gap as can be seen between the upper bound of Rule 1 and the lower bound of Rule 2 and thus the merging cannot be performed.

Figure 5.3 shows another example of three rules sharing the same two attributes α_1 and α_2 (terms) and referring to the same class label (high). While Rule 1 and Rule 3 are fully overlapped in both terms (T1 and T2), Rule 2 is partially overlapped with them by only a single term (T1).



Rule 1: if (3.5 < $\alpha_1 \leq$ 8.6) & (10.6 < $\alpha_2 \leq$ 14.7) Then high
Rule 2: if (6.8 < $\alpha_1 \leq$ 12.9) & (16.1 < $\alpha_2 \leq$ 22.9) Then high
Rule 3: if (2.8 < $\alpha_1 \leq$ 11.3) & (11.3 < $\alpha_2 \leq$ 13.2) Then high

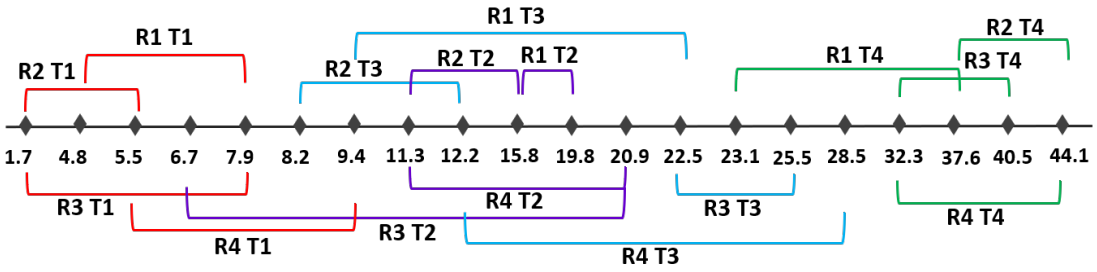
Figure 5.3: Rule set with two rule-terms sharing similar features and classes (before merging)

As a result, Rule 2 cannot be incorporated in the merging process due to the gap existing in (T2) between 14.7 and 16.1 whereas Rule 1 and Rule 3 can be combined into a single rule. The final output of this process is the rule set shown in Rules 5.2

$$\left.\begin{array}{r}\textbf{Merged Rule: if (2.8 < } \alpha_1 \leq \textbf{ 11.3) \& (10.6 < } \alpha_2 \leq \textbf{ 14.7) Then high} \\ \textbf{Rule 2: if (6.8 < } \alpha_1 \leq \textbf{ 12.9) \& (16.1 < } \alpha_2 \leq \textbf{ 22.9) Then high}\end{array}\right\} \quad (5.2)$$

Rules 5.2: A rule set produced after applying the local RM approach to the example of rules presented in Figure 5.3.

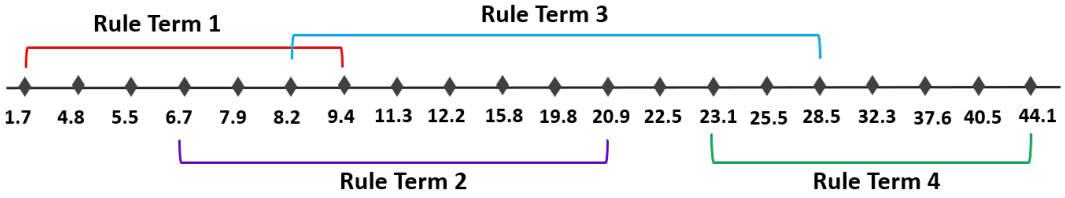
As previously stated, the main advantage of this approach is reducing the complexity and improving the interpretability of rules that might be generated from large datasets or high dimensional data. Consequently, the number of rules for each selected base classifier in the ensemble model would be reduced by removing the overlap that might occur between rules and thus also reduce the computational cost of prediction. Following is another example to show how beneficial this Rule Merging can be. Figure 5.4 includes four rules (Rule 1, Rule 2, Rule 3, Rule 4); each of which shares the same attributes ($\alpha_1, \alpha_2, \alpha_3, \alpha_4$) and refers to the same class label (low).



Rule 1: if (4.8 < $\alpha_1 \leq$ 7.9) & (15.8 < $\alpha_2 \leq$ 19.8) & (9.4 < $\alpha_3 \leq$ 22.5) & (23.1 < $\alpha_4 \leq$ 37.6) Then low
Rule 2: if (1.7 < $\alpha_1 \leq$ 5.5) & (11.3 < $\alpha_2 \leq$ 15.8) & (8.2 < $\alpha_3 \leq$ 12.2) & (37.6 < $\alpha_4 \leq$ 44.1) Then low
Rule 3: if (1.7 < $\alpha_1 \leq$ 7.9) & (6.7 < $\alpha_2 \leq$ 20.9) & (22.5 < $\alpha_3 \leq$ 25.5) & (32.3 < $\alpha_4 \leq$ 40.5) Then low
Rule 4: if (5.5 < $\alpha_1 \leq$ 9.4) & (11.3 < $\alpha_2 \leq$ 20.9) & (12.2 < $\alpha_3 \leq$ 28.5) & (32.3 < $\alpha_4 \leq$ 44.1) Then low

Figure 5.4: Rule set with multiple rule-terms sharing similar features and classes (before merging)

Assume that a classifier is searching this rule set to find the first rule that covers an instance with the following attributes values: ($\alpha_1 = 8.1,\ \alpha_2 = 20.2,\ \alpha_3 = 27.5,\ \alpha_4 = 43.4$). In this case, the classifier is required to check the whole rule set to find the first match which is Rule 4. Figure 5.4 shows that each rule-term in any of the rules is either completely or partially overlapped with at least one rule that includes the same attribute. Hence, applying the rule merging method to this rule set, as illustrated in Figure 5.5, replaces the four rules with a single merged rule and thus less effort is required to find a rule that matches the instance.



if (1.7 < $\alpha_1 \leq$ 9.4) & (6.7 < $\alpha_2 \leq$ 20.9) & (8.2 < $\alpha_3 \leq$ 28.5) & (23.1 < $\alpha_4 \leq$ 44.1) Then low

Figure 5.5: A rule set with 4 rule-terms produced after merging the rules listed in Figure 5.4

5.3.5 Stage 5: Combination and Prediction

The underlying philosophy of ensemble systems, which was described in Section 2.5.1 states that instead of trying to determine the perfect single model to provide a correct prediction for a new unseen instance, ensemble methods can combine the predictions of a diverse set of models in order to achieve accurate prediction ability. However, it is essential to an ensemble combiner to utilise the appropriate combination strategy in order to produce not only accurate but also more robust classification results [73]. As clarified in Section 5.3.1, ReG-Rules adopts the parallel learning strategy, i.e. each base classifier in the ensemble can be independently created in parallel to other base classifiers' constructions without cooperation in the training phase. Instead, collaborations between them can exist only in testing phase where their individual predictions are passed to a combiner using a combination strategy [92].

In Chapter 2 (Section 2.6.3), a number of combination methods have been reviewed. Amongst these were the two frequently used strategies: (1) majority voting and (2) weighted voting. Regarding (1), all the base models have the same weights and thus in the testing step, the ensemble classifier will assign an unlabelled instance to the class that has the highest number of votes. A number of widely used ensemble classifiers such as Random Forest [101] use this equal voting method to decide the final classification output. With respect to (2) and unlike the majority voting, weighted voting combination method considers the performance evaluation of each base classifier on a validation set as a weight to avoid a potential problem of reliability that may occur in simple majority voting method when some base classifiers are more reliable than others. Hence, assigning higher weights to the decisions of those qualified models may further improve the overall predictive performance than can be achieved by the equal majority voting [24, 35].

ReG-Rules Combination Approach

The combination method adopted in this research is based on the weighted voting strategy, but not just on classifier level but also on individual rule level. For this, ReG-Rules builds a committee of rules termed *Classification Committee* as can be seen in Figure 5.1. The process is detailed in Algorithm 9. In the algorithm, i refers to the new unseen instance, T denotes the test data, and n is the number of the top-ranked base classifiers ($topBC$). Essentially for each i, the combiner creates a committee of rules, which comprises the first rule that fired from each $topBC$ as seen in Figure 5.1.

Algorithm 9: Combiner: ReG-Rules Committees

Notations: i: an unseen instance, T: Test data, n: number of top-ranked base classifiers, j: a base classifier, C: class label, R: rule set, w: weight, $topBC$: top-ranked base classifier

1 **for** $i = 1 \rightarrow T$ **do**
2 Generate new classifier committe $comm$
3 **for** $j = 1 \rightarrow\ n$ *($topBC$ selected and weighted in Algorithm 6)* **do**
4 $vote_j \leftarrow$ predict class C_i for instance i
5 Add $vote_j$ to $comm_i$ including the weight of the model $topBC_j$ and the weight of its rule set R_j that has been used for the prediction
6 **end**
7 Eliminate the abstaining classifiers whose rule set does not cover the instance i
8 Compute the score w_i for each class in $comm_i$
9 **return** *committe decision $comm_i$ that has highest weighted average probability*
10 Evaluate $comm_i$ final prediction
11 **end**

Table 5.1 shows this committee of rules on an example, how it has been computed by lines 3 to 6 in Algorithm 9. Each prediction received by the committee from the $topBC$ is associated with the following components:

1. Tentative accuracy of the base classifier from which the rule comes from. The Tentative accuracy is computed only on classification attempts.

2. The number of times a rule was used during the validation phase and predicted the correct class label (CUR).

3. The predicted class label of the rule.

4. The classification type, i.e. did the base classifier use a rule or was it just a majority vote.

Next, in lines 7 to 10 in Algorithm 9 the votes are combined. First all votes that are based on majority class as classification type are not considered for computing the weight. The reason is that no rule has fired for these base classifiers, thus they have abstained, and their votes are considered unreliable. Then, the scores for each class label in Table 5.1 are computed, in this case there are 3 class labels namely A, B, and C.

Table 5.1: Example of metrics contained in a committee of 20 rules for the classification of one test instance

Classifier No.	Rule ID	CUR times	Tentative Acc.	Vote	Classification Type
34	8	3	1.0	Class B	Rules
14	8	0	1.0	Class A	Rules
80	3	10	1.0	Class B	Rules
54	4	12	1.0	Class B	Rules
25	12	3	1.0	Class B	Rules
84	-	-	1.0	Class C	Majority class
20	3	12	1.0	Class B	Rules
59	10	0	1.0	Class A	Rules
77	4	7	1.0	Class B	Rules
12	3	12	1.0	Class B	Rules
38	-	-	1.0	Class C	Majority class
7	10	0	1.0	Class A	Rules
53	3	9	1.0	Class B	Rules
71	4	7	1.0	Class B	Rules
81	4	3	1.0	Class B	Rules
60	12	1	0.94	Class B	Rules
50	12	0	0.93	Class B	Rules
90	7	2	0.91	Class B	Rules
73	13	3	0.85	Class A	Rules
46	10	2	0.75	Class C	Rules

The computed scores in this example are shown in Table 5.2. For each predicted class, the score contains the following components:

1. Vote frequency, which is simply how often there is a base classifier in the classification committee that voted for a particular class.

2. Sum of tentative accuracies of the rule sets' base classifiers that have voted for that class.

3. Total CUR, which is the sum of all CUR values of the rules' base classifiers that voted for that class.

Hence, as it can be seen in Table 5.2:

$$\text{Total CUR for class A} = 3$$
$$\text{Total CUR for class B} = 81$$
$$\text{Total CUR for class C} = 2$$

Table 5.2: Predicted Classes' Scores

Predicted class	Vote Frequency	Total CUR per class	Sum Ten. Acc. per class
Class A	4	3	3.85
Class B	13	**81**	12.78
Class C	1	2	0.75

Accordingly, CUR value is used to assign a class to the test instance for which the committee of rules was build for. A higher CUR indicates a better class label discrimination, and thus is selected as the final prediction of the committee. If there is a tie-break, meaning two or more classes have achieved the same highest CUR, then the highest sum of tentative accuracies per class is used to discriminate further. If the tie-break issue still exist, then vote frequency per class label is considered.

5.4 Evaluation

The aims of the experiments in this chapter are:

1. To empirically evaluate the overall performance of the new rule-based ensemble learner ReG-Rules compared with the stand-alone classifier (G-Rules-IQR), and also explores its runtime complexity.

2. To empirically evaluate the performance of these two new methods, which are integrated in the ensemble ReG-Rules learner:

    - Ranking-based CUR approach, which was proposed in Section 5.3.3, and used as a criterion to rank the base classifiers before selecting the top models according to a predefined size.
    - Rule Merging (RM) approach, which was proposed in Section 5.3.4, and used to improve the quality of the rules for each base classifier in the ensemble locally and independently.

5.4.1 Experimental Setup

All the experiments were performed on a 2.9 GHz Quad-Core Intel Core i7 machine with 16 GB 2133 MHz LPDDR3, running macOS Catalina version 10.15.1. All the 24 datasets used in the experiments were chosen randomly from the UCI repository [17], the only condition being that they contain continuous attributes and involve classification tasks. The specifications of the datasets are highlighted in Table 5.3 in terms of number of instances, attributes (including the type of attributes) and class labels. Datasets 15, 16 and 24 included few missing values in continuous attributes. To address this issue, the current research adopted and implemented a common method, found in the literature [5], which is based on estimating a missing numeric value with the *average value* for the concerning attribute.

Table 5.3: Characteristics of the datasets used in Chapter 5 experiments

No.	Dataset	No. Attributes	No. Classes	No. Instances
1.	iris	5 (4 cont)	3	150
2.	seeds	8 (7 cont)	3	210
3.	wine	14 (13 cont)	3	178
4.	blood transfusion	6 (5 cont)	2	748
5.	banknote	6 (5 cont)	2	1,372
6.	ecoli	9 (7 cont, 1 name)	8	336
7.	yeast	10 (8 cont, 1 name)	10	1,484
8.	page blocks	11 (10 cont)	5	5,473
9.	user modelling	6 (5 cont)	4	403
10.	breast tissue	11 (10 cont)	6	106
11.	glass	11 (10 cont, 1 id)	7	214
12.	HTRU2	10 (9 cont)	2	17,898
13.	magic gamma	12 (11 cont)	2	19,020
14.	wine quality-white	13 (12 cont)	11	4,898
15.	breast cancer	12 (10 cont, 1 id)	2	699
16.	post operative	10 (1 cont, 9 categ)	3	90
17.	wifi localization	8 (7 cont)	4	2,000
18.	indian liver patient	12 (10 cont, 1 categ)	2	583
19.	sonar	62 (61 cont)	2	208
20.	leaf	17 (15 cont, 1 name)	40	340
21.	internet firewall	12 (cont)	4	65,532
22.	bank marketing	17 (6 cont, 10 categ)	2	45,211
23.	avila	11 (10 cont)	12	20,867
24.	shuttle	10 (9 cont)	7	58,000

Moreover, all algorithms have been implemented in the statistical programming language R [147] and reuse the same code base, differing only in the methodological aspects described in this chapter. The algorithms were evaluated against 5 metrics for classifiers, which are presented as follows:

1. Number of Rules: this is the total number of rules generated by G-Rules-IQR classifier and the average rules generated by the ensemble base classifiers.

2. F1 score: this is also known as the harmonic mean of precision and recall. This is a number between 0 and 1. A high F1 score is desired.

3. Accuracy: this is the ratio of instances that have been correctly classified, either using the induced rule set, or the majority class strategy in case of unclassified instances. This is a number between 0 and 1.

4. Tentative Accuracy: this is the ratio of instances that have been correctly classified using only the induced rule set, i.e. does not count instances that not covered by rules.

5. Abstaining Rate: this is the proportion of cases a classifier abstains from classification, i.e. the proportion of instances not covered by the rule set. This is a number between 0 and 1. A low abstaining rate is desired.

It is important to note, that tentative accuracy does not consider the abstained instances, while the accuracy counts them as misclassifications. Hence, the higher the abstaining rate, the higher the tentative accuracy and the lower the accuracy.

Regarding the empirical evaluation of the ensemble ReG-Rules classifier in comparison with its stand-alone G-Rules-IQR classifier, the following two evaluation methods have been applied to all the experimental datasets presented in this chapter:

1. **Separate Train and Test Datasets**. In this evaluation method, each dataset was *randomly sampled without replacement* into train and test datasets. While the 70% of the data instances were used to train and build the ensemble classifier, the remaining 30% were used as a testing dataset.

2. **Five-fold Cross Validation**. In this evaluation method, each dataset was shuffled and randomly divided into 5 partitions (folds) of equal size. Then for each fold, a learning algorithm was trained on the remaining four folds and then tested on the current fold.

In both evaluation methods, the test set is used only once to assess the general performance of the classification models. In case of ReG-Rules model, the train dataset is used to generate multiple base classifiers using bagging, which is a method of sampling with replacement that can be used to ensure a sufficient level of diversity during the base classifiers' generation phase.

5.4.2 Runtime Complexity of the Ensemble ReG-Rules Classifier

The induction process of ReG-Rules involves 4 components that need to be considered for estimating the runtime complexity with respect to the number of instances N and the number of features M. These components are (1) Diversity Generation, (2) Base Classifier Inductions, (3) Models Selection and (4) Rules Merging (see Figure 5.1). These components are executed sequentially, hence the complexity of ReG-Rules is determined by the component with the highest complexity [22].

With respect to component (1) Bagging is used. Bagging has a complexity of $O(N)$ since N sample instances are taken. Bagging is not dependent on the number of features M. Hence, the complexity of Diversity Generation can be described with $O(N)$. With respect to component (2), G-Rules-IQR base classifiers have a theoretical worst case complexity of $O(N^2M)$ [21], in which case each data instance would be covered by a single rule. However, according to [110] the complexity of algorithms of the PRISM family (to which G-Rules-IQR belongs) is estimated to be linear on average. Thus, the runtime complexity of ReG-Rules' Base

Classifier Inductions component can be estimated as $d \cdot O(N^2 \cdot M)$, where d is the number of base classifiers induced. d can be neglected here as it is not dependent on the size of the training data. With respect to component (3) the Models Selection is not dependent on the training data but represents a summand m added to the runtime dependent only on the number of models generated. With respect to component (4) the merging of rules is not dependent on the training data but represents a summand r depending only on the number of rules generated. Thus, the total runtime complexity can be described as $O(N) + O(N \cdot M) + m + r$, which can be simplified to a runtime complexity of $O(N \cdot M)$. Thus, it can be said that ReG-Rules is likely to scale linearly with respect to the number of data instances and features in the training data.

5.4.3 Empirical Evaluation of the Ensemble ReG-Rules Classifier

The experimental results presented in Tables 5.4 and 5.5 were obtained using the train and test evaluation method. Similarly, Tables 5.6 and 5.7 present the experimental results acquired using the five-fold cross validation method. Each evaluation method's results will be discussed separately in the following sub-sections. In each table, the # symbol refers to the index of the dataset in Table 5.3. The best result(s) in the tables for each dataset are highlighted in bold letters. The tables show the results with respect to the 5 evaluation metrics previously described in Section 5.4.1.

5.4.3.1 Evaluation Using Separate Training and Testing Dataset Strategy

The number of rules and the abstaining rates computed using train and test strategy are shown in Table 5.4. Regarding 'number of rules' metric, three types of induced rule sets are compared for each dataset: (1) number of rules generated by the stand-alone G-Rules-IQR classifier, (2) average number of rules induced by the ensemble ReG-Rules classifier before utilising the local RM algorithm, and (3) average number of rules generated by ReG-Rules after integrating the local RM algorithm in its selected base classifiers' rule sets. As can be seen in the table, on average, a ReG-Rules base classifier produces fewer rules than G-Rules-IQR for all the 24 datasets.

However, further reduction in the number of induced rules without reducing the performance of the base classifier is desired and beneficial to the human analyst. For this reason, ReG-Rules integrates the local RM approach in its construction. The results presented in Table 5.4 demonstrates a considerable reduction

in the number of rules generated by ReG-Rules classifier after applying the local RM method, i.e. particularly in 18 out of 24 datasets. As shown in Figure 5.6, in some cases the reduction was more than 45%. In 6 out of 24 datasets ReG-Rules classifier produced the same number of rule sets before and after utilising the RM method. Importantly, and due to this significance, the remaining experimental results in this chapter will consider only the version of ReG-Rules, which employs the local RM technique in its construction.

Concerning the second part of Table 5.4, the abstaining rate metric results show that ReG-Rules achieves the lower rate in all the 24 cases. On 21 datasets ReG-Rules classifier lowers the abstaining rate to zero and on the remaining 3 cases its abstaining rate was very close to zero. While in terms of the stand-alone base classifier, G-Rules-IQR, the abstaining rates were higher than 10% on several datasets. In three datasets (9, 10, and 20) G-Rules-IQR's abstaining rate reaches 30%, 19% and 40% respectively.

Table 5.4: Number of Rules and Abstaining Rates using separate training and testing sets method

#	Number of Rules			Abstaining Rate	
	G-Rules-IQR	ReG-Rules		G-Rules-IQR	ReG-Rules
		before merging	after merging		
1	18	17	**13**	0.07	**0.00**
2	22	19	**15**	0.03	**0.00**
3	13	13	**11**	0.06	**0.00**
4	20	16	**11**	**0.00**	**0.00**
5	89	**82**	**82**	0.02	**0.00**
6	37	**32**	**32**	0.02	**0.00**
7	99	**82**	**82**	0.04	**0.00**
8	131	115	**104**	0.02	**0.00**
9	57	45	**42**	0.30	**0.00**
10	28	24	**23**	0.19	**0.00**
11	30	25	**22**	0.11	**0.02**
12	31	26	**17**	**0.00**	**0.00**
13	79	69	**50**	**0.00**	**0.00**
14	126	115	**62**	0.02	**0.00**
15	11	9	**8**	**0.00**	**0.00**
16	29	**23**	**23**	0.11	**0.00**
17	59	48	**38**	0.01	**0.00**
18	47	**50**	**50**	0.17	**0.00**
19	16	13	**12**	0.13	**0.00**
20	124	101	**98**	0.40	**0.01**
21	21	20	**16**	**0.00**	**0.00**
22	40	**38**	**38**	0.01	**0.00**
23	158	164	**146**	0.06	**0.01**
24	27	21	**19**	**0.00**	**0.00**

Table 5.5 uses the evaluation method of separate Training and Test datasets to compare ReG-Rules and G-Rules-IQR classifiers in terms of *F1 score*, *accuracy* and *tentative accuracy* metrics. With respect to F1 score, which is the harmonic mean of precision and recall, the results show that the proposed ReG-Rules outperforms

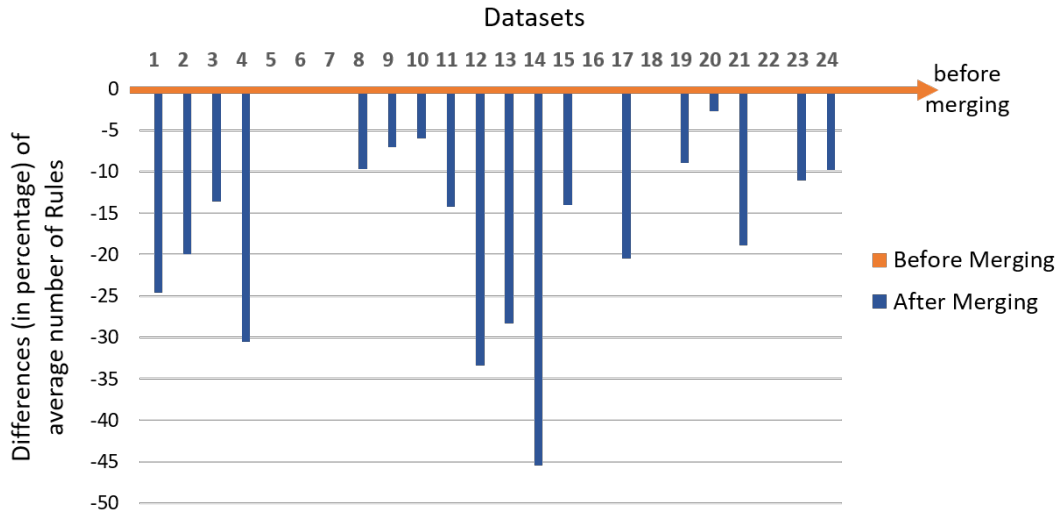


Figure 5.6: Difference (in percentage) of average number of rules of ReG-Rules classifier after integrating RM approach compared with before the merging process

G-Rules-IQR on 12 out of 24 datasets. Moreover, in 6 out of the remaining 12 cases where ReG-Rules did not outperform its competitor, it still performs at the same level of F1 score as G-Rules-IQR. On two datasets (1 and 9), ReG-Rules algorithm was not the best method, but was still very close within 3% difference to the best F1 score.

Table 5.5: F1 score, General Accuracy and Tentative Accuracy using separate training and testing sets method

#	F1 score		Accuracy		Tentative Accuracy	
	G-Rules-IQR	ReG-Rules	G-Rules-IQR	ReG-Rules	G-Rules-IQR	ReG-Rules
1	**0.96**	0.93	0.91	**0.93**	**0.95**	0.93
2	**1.00**	**1.00**	0.97	**1.00**	**1.00**	**1.00**
3	0.98	**1.00**	0.94	**1.00**	0.98	**1.00**
4	0.98	**1.00**	0.97	**1.00**	0.97	**1.00**
5	0.98	**0.99**	0.98	**0.99**	**0.99**	**0.99**
6	**0.99**	0.91	**0.96**	0.95	**0.97**	0.95
7	0.87	**0.91**	0.93	**0.97**	0.96	**0.97**
8	**0.93**	0.87	**0.98**	**0.98**	**0.99**	0.98
9	**0.96**	0.95	0.72	**0.94**	**0.95**	0.94
10	0.81	**0.97**	0.66	**0.97**	0.81	**0.97**
11	0.86	**0.93**	0.86	**0.95**	**0.97**	**0.97**
12	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
13	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
14	**0.96**	0.87	0.97	**1.00**	0.99	**1.00**
15	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
16	0.52	**0.77**	**0.67**	0.63	**0.67**	0.63
17	**1.00**	**1.00**	0.99	**1.00**	**1.00**	**1.00**
18	0.83	**0.84**	0.71	**0.73**	0.71	**0.73**
19	0.95	**0.97**	0.87	**0.97**	0.94	**0.97**
20	**0.75**	0.67	0.39	**0.56**	**0.65**	0.57
21	**0.90**	**0.90**	**1.00**	**1.00**	**1.00**	**1.00**
22	**0.99**	**0.99**	**0.98**	**0.98**	**0.98**	**0.98**
23	0.89	**0.93**	0.85	**0.92**	0.88	**0.92**
24	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

In terms of accuracy, Table 5.5 shows that, in almost all cases, ReG-Rules achieved the highest accuracy. In particular, it outperforms the G-Rules-IQR algorithm in 15 out of 24 datasets and performs at the same level as its competitor in 7 out of the 9 remaining cases. On one of the two datasets (16) where ReG-Rules did not outperform G-Rules-IQR, the accuracy of ReG-Rules was lower by only 4% compared with G-Rules-IQR. With respect to tentative accuracy, the ReG-Rules algorithm performs better or equal than G-Rules-IQR in 18 out of 24 datasets. In these 18 cases, the proposed ReG-Rules classifier outperforms G-Rules-IQR in 8 cases. On 5 out of the remaining 6 cases, where ReG-Rules was not the best method, it only underperformed by a maximum difference of 3%.

5.4.3.2 Evaluation Using Cross Validation Strategy

The number of the rules generated by the G-Rules-IQR classifier is compared with the average number of rules generated by the ensemble ReG-Rules classifier that integrated local RM approach in its construction.

Table 5.6: Number of Rules and Abstaining Rates using cross validation method

#	Number of Rules		Abstaining Rate	
	G-Rules-IQR	ReG-Rules	G-Rules IQR	ReG-Rules
1	20	**14**	0.10	**0.00**
2	26	**17**	0.08	**0.00**
3	13	**11**	0.07	**0.00**
4	21	**11**	0.01	**0.00**
5	94	**81**	0.01	**0.00**
6	43	**36**	0.08	**0.00**
7	111	**92**	0.05	**0.00**
8	135	**107**	0.02	**0.00**
9	72	**50**	0.09	**0.00**
10	30	**24**	0.29	**0.01**
11	30	**22**	0.12	**0.00**
12	31	**17**	**0.00**	**0.00**
13	83	**57**	**0.00**	**0.00**
14	132	**66**	0.02	**0.00**
15	10	**8**	0.01	**0.00**
16	34	**25**	0.09	**0.00**
17	59	**39**	0.02	**0.00**
18	51	**51**	0.18	**0.00**
19	17	**13**	0.12	**0.00**
20	138	**107**	0.42	**0.00**
21	23	**19**	0.01	**0.00**
22	41	**39**	0.02	**0.00**
23	160	**143**	0.11	**0.01**
24	26	**20**	**0.00**	**0.00**

The results are demonstrated in Table 5.6 in which on average a ReG-Rules base classifier produces fewer rules than G-Rules-IQR on all the 24 datasets. The table also shows that compared with stand-alone G-Rules-IQR, the abstaining rate

in ReG-Rules was almost zero on all 24 datasets. Only on two cases (10 and 23) was ReG-Rules' abstaining rate slightly above zero.

Table 5.7: F1 score, General Accuracy and Tentative Accuracy using cross validation method

#	F1 score		Accuracy		Tentative Accuracy	
	G-Rules-IQR	ReG-Rules	G-Rules-IQR	ReG-Rules	G-Rules-IQR	ReG-Rules
1	0.95	**0.96**	0.88	**0.97**	0.96	**0.97**
2	**1.00**	**1.00**	0.93	**1.00**	**1.00**	**1.00**
3	0.99	**1.00**	0.92	**1.00**	0.99	**1.00**
4	0.98	**1.00**	0.96	**1.00**	0.97	**1.00**
5	**0.97**	**0.97**	0.96	**0.97**	0.96	**0.97**
6	0.92	**0.93**	0.87	**0.95**	0.93	**0.95**
7	**0.90**	0.89	0.93	**0.97**	**0.98**	0.97
8	**0.90**	0.86	0.98	**0.98**	**0.99**	0.98
9	0.95	**0.96**	0.89	**0.96**	0.95	**0.96**
10	**0.94**	0.90	0.70	**0.88**	**0.93**	0.88
11	**0.93**	0.86	0.86	**0.93**	**0.96**	0.93
12	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
13	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
14	**0.96**	**0.96**	0.98	**1.00**	**1.00**	**1.00**
15	**1.00**	**1.00**	0.99	**1.00**	**1.00**	**1.00**
16	0.69	**0.76**	0.66	**0.69**	0.67	**0.69**
17	0.99	**1.00**	0.98	**1.00**	0.99	**1.00**
18	**0.83**	**0.83**	0.70	**0.71**	**0.72**	0.71
19	0.96	**0.97**	0.89	**0.97**	0.96	**0.97**
20	**0.80**	0.70	0.37	**0.55**	**0.61**	0.55
21	0.84	**0.85**	0.99	**1.00**	**1.00**	**1.00**
22	**0.99**	**0.99**	0.97	**0.98**	**0.98**	**0.98**
23	0.93	**0.94**	0.87	**0.93**	**0.92**	**0.92**
24	**0.99**	0.98	**1.00**	**1.00**	**1.00**	**1.00**

Table 5.7 compares the ensemble ReG-Rules classifier and the stand-alone G-Rules-IQR classifier in terms of F1 score, accuracy and tentative accuracy using the evaluation method of 5-fold cross validation.

Regarding F1 score, the results illustrate that ReG-Rules achieves best F1 score on 18 out of 24 datasets. On these 18 datasets, ReG-Rules was the best classifier in 10 cases and performs at the same level of F1 scores as its competitor in the remaining 8 cases. Also, on 4 out of the remaining 6 datasets (7, 8, 10, 24), ReG-Rules classifier only underperformed by a maximum difference of 4%.

With respect to accuracy, ReG-Rules classifier outperforms G-Rules-IQR in 21 out of 24 datasets and performs at the same level on the remaining 3 datasets. Regarding tentative accuracy, ReG-Rules performs equal or better than G-Rules-IQR on 18 out of 24 datasets. Among these 18 cases, ReG-Rules outperforms G-Rules-IQR on 9 datasets.

5.4.3.3 Evaluation Summary of the Ensemble ReG-Rules Classifier

Generally speaking, the results of all the experiments conducted in this research using cross validation are consistent with the results obtained from the evaluation strategy of separate training and testing datasets. Both strategies show that ReG-Rules base classifiers are not only producing fewer rules on all the 24 datasets, but also almost never abstain from making a classification decision compared with G-Rules-IQR, which suffers from high abstaining rate on multiple datasets. In terms of F1 score, accuracy, and tentative accuracy, both evaluation approaches demonstrate that ReG-Rules outperforms G-Rules-IQR in most cases.

5.4.4 Empirical Evaluation of Ranking-based CUR Approach

As previously explained in Section 5.3.2, CUR stands for Correctly Used Rules, which is the number of times a rule was used to cover instances during the validation stage and predicted the correct class label. It is a 'track record prediction' for each rule separately in each individual base classifier. In other words, CUR is a rule's quality measurement, which will be associated with the rule and also be part of the general performance weighing of the base classifier that has generated this rule. The ensemble ReG-Rules system uses the CUR values of a certain model to compute the average CUR value of that model. Then as presented in Section 5.3.3 ReG-Rules ranks once its individual members according to not only their overall accuracy but also their CUR values and in case of ties the ranking will be based on the ascending order of their abstaining rates. Consequently, ReG-Rules system selects the top base classifiers whose rank is above a given threshold (either a fixed user defined amount or percentage of models).

In this part of the evaluation section, the ranking-based CUR approach is empirically evaluated in order to show not only its performance but also to what extent this strategy contributes towards the improvement of overall accuracy of the ensemble classification. For evaluation purposes, there are two versions of the ensemble ReG-Rules implemented using the same code base but differing in the ensemble selection method:

- **Version 1.** ReG-Rules learner incorporates a prior ranking to its base classifiers according to the ranking-based CUR method before selecting the top-ranked members.
- **Version 2.** ReG-Rules learner that does not involve any ranking process to its composite classifiers before selecting the same subset size of ensemble as for the first version.

Table 5.8: Comparison between two types of ensemble selection models applied to ReG-Rules classifier in terms of number of rules and abstaining rate

#	Number of Rules		Abstaining Rate	
	No Ranking	Ranking	No Ranking	Ranking
1	**11**	13	**0.00**	**0.00**
2	16	**15**	**0.00**	**0.00**
3	**11**	**11**	**0.00**	**0.00**
4	16	**11**	**0.00**	**0.00**
5	**80**	82	**0.00**	**0.00**
6	**31**	32	**0.00**	**0.00**
7	**82**	**82**	**0.00**	**0.00**
8	**101**	104	**0.00**	**0.00**
9	**42**	**42**	**0.00**	**0.00**
10	**22**	23	**0.00**	**0.00**
11	**21**	22	**0.00**	0.02
12	20	**17**	**0.00**	**0.00**
13	90	**50**	**0.00**	**0.00**
14	64	**62**	**0.00**	**0.00**
15	**8**	**8**	**0.00**	**0.00**
16	**21**	23	**0.00**	**0.00**
17	**36**	38	**0.00**	**0.00**
18	51	**50**	**0.00**	**0.00**
19	**12**	**12**	**0.00**	**0.00**
20	**97**	98	**0.00**	0.01

Table 5.9: Comparison between two types of ensemble selection models applied to ReG-Rules classifier in terms of F1 score, Accuracy and Tentative Accuracy

#	F1 score		Accuracy		Tentative Accuracy	
	No Ranking	Ranking	No Ranking	Ranking	No Ranking	Ranking
1	0.91	**0.93**	0.91	**0.93**	0.91	**0.93**
2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
3	0.98	**1.00**	0.98	**1.00**	0.98	**1.00**
4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
5	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
6	**0.91**	**0.91**	0.94	**0.95**	0.94	**0.95**
7	0.89	**0.91**	**0.97**	**0.97**	**0.97**	**0.97**
8	0.85	**0.87**	0.97	**0.98**	0.97	**0.98**
9	0.93	**0.95**	0.93	**0.94**	0.93	**0.94**
10	0.87	**0.97**	0.88	**0.97**	0.88	**0.97**
11	0.90	**0.93**	**0.97**	0.95	**0.97**	**0.97**
12	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
13	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
14	**0.87**	**0.87**	**1.00**	**1.00**	**1.00**	**1.00**
15	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
16	0.74	**0.77**	0.59	**0.63**	0.59	**0.63**
17	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
18	0.83	**0.84**	0.71	**0.73**	0.71	**0.73**
19	0.96	**0.97**	0.95	**0.97**	0.95	**0.97**
20	0.66	**0.67**	**0.57**	0.56	**0.57**	**0.57**

The results of the experiments are detailed in Tables 5.8 and 5.9 using the five 5 metrics described in Section 5.4.1. The best result(s) in these tables for each dataset are highlighted in bold letters. With regard to number of rules measure, the results demonstrated in Table 5.8 suggests that the ranked version of ReG-

Rules achieves the best results in 11 out of 20 datasets. In terms of abstaining rates metric, both versions performed almost equally well.

The results demonstrated in Table 5.9 show that integrating the ranking method into the proposed ensemble algorithm improves the classification performance in most cases in terms of F1 score, accuracy, and tentative accuracy.

Regarding F1 score, the table shows that ReG-Rules (version 1)[2] outperforms (version 2)[3] in 11 out of 20 datasets. Also, among the remaining 9 cases where it is not surpassing, the ranked version of ReG-Rules algorithm achieves similar F1 scores on 8 datasets compared with the other version.

Concerning accuracy, ReG-Rules (version 1) achieves the highest results in 18 out of 20 datasets. Only on two datasets (11 and 20) where the proposed ReG-Rules algorithm was at most 2% lower in accuracy than the results accomplished by ReG-Rules (version 2).

In terms of tentative accuracy, ReG-Rules (version 1) performs equal or better than the other version of ReG-Rules on all the 20 datasets.

For simplicity, the bar chart shown in Figure 5.7 summarises the performance of ReG-Rules with ranking-based CUR approach over ReG-Rules classifier without the ranking as follows:

- Number of rules: the results indicates that there is no clear winner with 6 wins and 5 ties for ReG-Rules (version 1).
- Abstaining rates: with 18 ties and only 2 losses out of 20, both versions of ReG-Rules are almost performed equally.
- F1 score and Tentative accuracy: there is no loss reported in these two metrics for ReG-Rules with ranking, i.e. it clearly outperformed its competitor.
- Overall accuracy: ReG-Rules with ranking reports 9 wins, 9 ties and only 2 losses. Thus, it outperformed version 2.

[2]ReG-Rules (version 1) incorporates a prior ranking to its base classifiers according to the ranking-based CUR method.

[3]ReG-Rules (version 2) does not involve any ranking process to its composite classifiers.

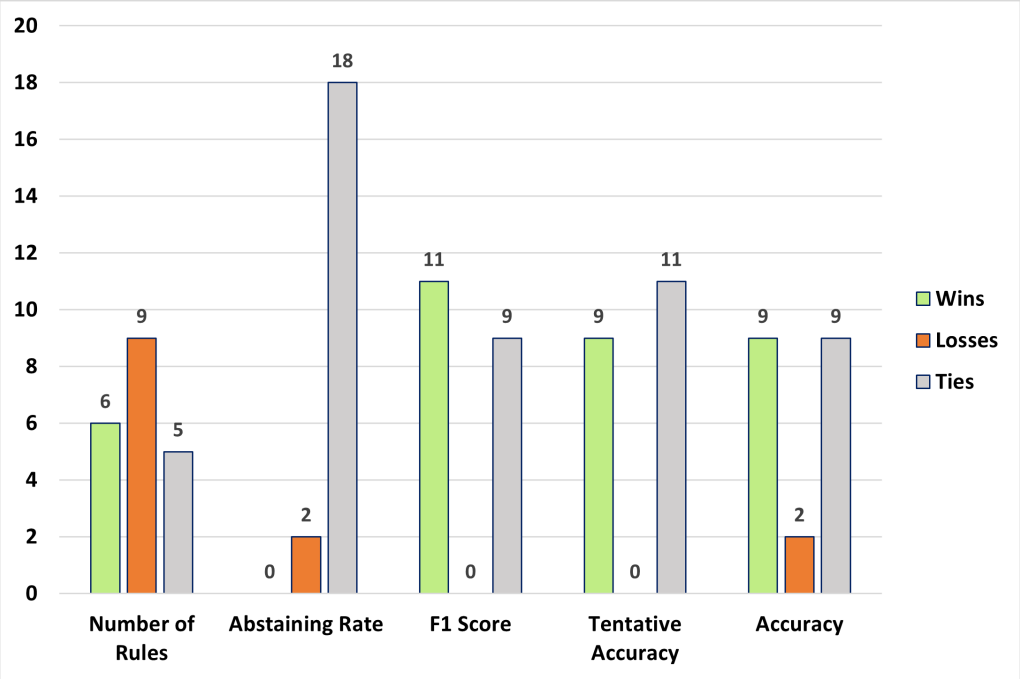


Figure 5.7: Performance of ReG-Rules classifier with ranking-based approach over ReG-Rules classifier without ranking-based approach

5.4.5 Qualitative Evaluation of Rules Merging (RM) Algorithm

Rule Merging (RM) method was proposed in Section 5.3.4 with the aim to mitigate the complexity of rule set for the individual classifier by reducing the number of rules and rule-terms. The RM approach has been integrated in the ensemble ReG-Rules classification model construction and was empirically evaluated in Section 5.4.3 with respect to the general performance of ReG-Rules classifier. This section evaluates the RM method qualitatively on two cases studies where the rule sets produced by a G-Rules-IQR classifier without RM and one with RM are examined.

The two case studies are the blood transfusion and the wine datasets from the UCI repository [17]. The descriptions of the two datasets can be found in Table 5.3 (3 and 4) in terms of number of instances, attributes (including type of attributes) and classes. Both datasets are used previously among other datasets to evaluate the original G-Rules-IQR algorithm in a published work [21]. The datasets have been randomly sampled without replacement into train and test datasets; whereas the test sets consist of 30% of the data instances and the remaining 70% were used to learn the rule set.

5.4.5.1 Case Study 1: Experiments Conducted on Blood Transfusion Dataset

The two rule sets listed in the following page are generated using the same 524 training instances derived from the same dataset (blood transfusion). The 20 original rules were induced by the base learning algorithm (G-Rules-IQR) before applying the RM approach, whereas the 12 merged rules were produced after utilising the rule merging technique. Both rule sets, the original and the merged, are evaluated using the same remaining 224 testing instances.

Table 5.10: Experimental results of Case Study 1

Metrics	Original rule set	Merged rule set
Number of Rules	20	**12**
Abstaining Rate	0	0
Recall	1	1
Precision	0.966	0.971
F1 score	0.982	0.985
Accuracy	0.973	0.977
Tentative Accuracy	0.973	0.977

The results are shown in Table 5.10 using 6 different metrics, and it can be seen that after applying the RM method, the number of rules is reduced from 20 to 12 rules (i.e. more than 40% reduction). This significant decrease in the rules size enhances the expressive power of the rule model and makes it easier for the analyst to understand the rules. By manually examining the two lists (original and merged), we can see that the RM approach merges without loss of information. Thus, instances covered by a rule before merging should still be covered either by the same rule or the resulting merged rule (leading to the same classification) after RM was applied.

With regard to precision, F1 score, accuracy and tentative accuracy, Table 5.10 shows very small variations in these metrics. A closer examination of the results of the test data revealed that the variations are a result of the order in which the rules are applied. Before merging a data instance may have been covered by two or more rules each leading to a different class label and the first rule applied and matching the data instance would determine the class label. The same effects are still true after the RM, if two rules are merged they are not any more listed consecutively and the rule order may change slightly.

Original Rules

R1: $18.59 < Time \leq 51.41 \rightarrow 0$

R2: $30.8 < Time \leq 73.2 \rightarrow 0$

R3: $2.41 < Monetary \leq 2.98$ &
$-1.06 < Time \leq 33.06$ &
$0.44 < Frequency \leq 0.51$ &
$0.70 < Recency \leq 1.01 \rightarrow 0$

R4: $2.41 < Monetary \leq 2.98$ &
$-2.44 < Time \leq 34.44$ &
$0.44 < Frequency \leq 0.51$ &
$0.50 < Recency \leq 0.90 \rightarrow 0$

R5: $2.41 < Monetary \leq 2.98$ &
$0.44 < Frequency \leq 0.51$ &
$1.45 < Time \leq 26.55 \rightarrow 0$

R6: $0.69 < Recency \leq 1.12$ &
$-3.45 < Time \leq 39.45$ &
$0.17 < Frequency \leq 1.44$ &
$2.39 < Monetary \leq 2.40 \rightarrow 0$

R7: $-6.43 < Time \leq 42.43$ &
$0.17 < Frequency \leq 0.43$ &
$0.93 < Recency \leq 1.42$ &
$2.39 < Monetary \leq 2.40 \rightarrow 0$

R8: $48.54 < Time \leq 99.46 \rightarrow 0$

R9: $6.60 < Time \leq 15.41 \rightarrow 0$

R10: $0.32 < Recency \leq 0.63$ &
$0.21 < Frequency \leq 0.39$ &
$1.99 < Time \leq 2.0 \rightarrow 0$

R11: $12.43 < Time \leq 19.57 \rightarrow 0$

R12: $3.99 < Time \leq 4.0 \rightarrow 0$

R13: $1.12 < Time \leq 1.64 \rightarrow 1$

R14: $0.75 < Time \leq 1.48 \rightarrow 1$

R15: $1.25 < Time \leq 2.03$ &
$0.87 < Frequency \leq 1.29 \rightarrow 1$

R16: $0.29 < Time \leq 1.11 \rightarrow 1$

R17: $1.76 < Time \leq 1.93 \rightarrow 1$

R18: $1.61 < Time \leq 1.82 \rightarrow 1$

R19: $1.60 < Frequency \leq 1.69 \rightarrow 1$

R20: $1.95 < Time \leq 1.97 \rightarrow 1$

Merged Rules

R1: $18.59 < Time \leq 99.46 \rightarrow 0$

R2: $2.41 < Monetary \leq 2.99$ &
$-2.44 < Time \leq 34.44$ &
$0.44 < Frequency \leq 0.51$ &
$0.50 < Recency \leq 1.10 \rightarrow 0$

R3: $0.69 < Recency \leq 1.42$ &
$-6.43 < Time \leq 42.43$ &
$0.17 < Frequency \leq 0.44$ &
$2.39 < Monetary \leq 2.40 \rightarrow 0$

R4: $6.6 < Time \leq 19.57 \rightarrow 0$

R5: $0.29 < Time \leq 1.64 \rightarrow 1$

R6: $1.61 < Time \leq 1.93 \rightarrow 1$
$0.21 < Frequency \leq 0.39$ &
$1.99 < Time \leq 2.0 \rightarrow 0$

R7: $2.41 < Monetary \leq 2.98$ &
$0.44 < Frequency \leq 0.51$ &
$1.45 < Time \leq 26.55 \rightarrow 0$

R8: $0.69 < Recency \leq 1.12$ &
$-3.45 < Time \leq 39.45$ &
$0.17 < Frequency \leq 1.44$ &
$2.39 < Monetary \leq 2.40 \rightarrow 0$

R9: $-6.43 < Time \leq 42.43$ &
$0.17 < Frequency \leq 0.43$ &
$0.93 < Recency \leq 1.42$ &
$2.39 < Monetary \leq 2.40 \rightarrow 0$

R10: $48.54 < Time \leq 99.46 \rightarrow 0$

R11: $6.60 < Time \leq 15.41 \rightarrow 0$

R12: $0.32 < Recency \leq 0.63$ &
$0.21 < Frequency \leq 0.39$ &
$1.99 < Time \leq 2.0 \rightarrow 0$

5.4.5.2 Case Study 2: Experiments Conducted on Wine Dataset

The two rule sets listed below are induced using the same 524 training instances sampled from the same dataset (wine). The 13 original rules were induced by the base learning algorithm (G-Rules-IQR) before applying the merging approach, while the 9 merged rules were produced after utilising the RM technique. Both rule sets, the original and the merged rule sets, are evaluated using the same test data examples, which consists of the remaining 53 instances. The results are shown in Table 5.11 using 6 different metrics, and it can be seen that after applying the RM method, the number of rules is lowered from 13 to 9, i.e. approx. 31% reduction in the rules size. As discussed for case study 1, a more compact rule set would make it more understandable by a human. Also, the merging does not cause loss of information, merely the rule order may be influenced. In this case, no effects of the rule order can be observed with respect to the performance metrics listed in Table 5.11.

Original Rules

R1: $0.09 < \textit{Noflavan phenols} \leq 0.12 \rightarrow 1$
R2: $0.58 < \textit{Total phenols} \leq 0.62 \rightarrow 1$
R3: $0.59 < \textit{Total phenols} \leq 0.65 \rightarrow 1$
R4: $0.05 < \textit{Noflavan phenols} \leq 0.11 \rightarrow 1$
R5: $13.68 < \textit{Alcohol} \leq 13.70 \rightarrow 1$
R6: $1.93 < \textit{Magnesium} \leq 2.02 \rightarrow 2$
R7: $1.85 < \textit{Magnesium} \leq 2.01 \rightarrow 2$
R8: $2.01 < \textit{Magnesium} \leq 2.15 \rightarrow 2$
R9: $2.77 < \textit{Proline} \leq 2.89 \rightarrow 2$
R10: $0.39 < \textit{Total phenols} \leq 0.46 \rightarrow 3$
R11: $509.6 < \textit{Proline} \leq 670.4 \rightarrow 3$
R12: $0.34 < \textit{Total phenols} \leq 0.43 \rightarrow 3$
R13: $0.57 < \textit{Hue} \leq 0.62 \rightarrow 3$

Merged Rules

R1: $0.05 < \textit{Noflavan phenols} \leq 0.12 \rightarrow 1$
R2: $0.58 < \textit{Total phenols} \leq 0.65 \rightarrow 1$
R3: $1.85 < \textit{Magnesium} \leq 0.65 \rightarrow 1$
R4: $0.34 < \textit{Total phenols} \leq 0.46 \rightarrow 1$
R5: $13.68 < \textit{Alcohol} \leq 2.02 \rightarrow 1$
R6: $2.01 < \textit{Magnesium} \leq 2.15 \rightarrow 2$
R7: $2.77 < \textit{Proline} \leq 2.89 \rightarrow 2$
R8: $509.6 < \textit{Proline} \leq 670.4 \rightarrow 3$
R9: $0.57 < \textit{Hue} \leq 0.62 \rightarrow 3$

Table 5.11: Experimental results of Case Study 2.

Metrics	Original rule set	Merged rule set
Number of Rules	13	**9**
Abstaining Rate	0.06	0.06
Recall	0.98	0.98
Precision	0.98	0.98
F1 score	0.98	0.98
Accuracy	0.94	0.94
Tentative Accuracy	0.98	0.98

5.5 Summary

The chapter presented the development of a new predictive ensemble learner termed ReG-Rules. ReG-Rules' purpose is to explore if it is possible to create an explainable predictive ensemble model while benefiting from predictive performance of ensemble learning. The chapter first identified G-Rules-IQR algorithm as the suitable candidate for the base learners' induction of ReG-Rules. This choice was because G-Rules-IQR has already been optimised in Chapter 4 to induce a highly expressive rule set and provides a high classification accuracy in comparison with other rule-based learners.

The chapter then further reviewed and discussed the limitations related to stand-alone learning systems that can be addressed by utilising ensemble learning methods. The general framework of the ReG-Rules model was introduced in Section 5.3, which consists of 5 stages: Diversity Generation, Base Classifiers Induction, Models Selection, Rules Improvement, and Combination and Prediction. Each component was illustrated in a different section.

ReG-Rules induces a diverse ensemble based on bagging. The induced base classifiers are ranked according to their classification performance on different validation datasets, and only the best performing classifiers are retained and considered for predicting class labels.

To measure the individual classification performance of these base classifiers, ReG-Rules uses a new weighting method composed of various metrics, not accuracy alone. Also, to rank and then select the best performing models, ReG-Rules uses a new Ranking-based method. Both integrated methods were developed and empirically evaluated in this chapter.

The rule sets of these best-ranked base classifiers are further optimised by merging overlapping rules within each rule set independently, and thus reducing the average number of rules in the base models. ReG-Rules utilises a new post-processing approach called 'local Rule Merging' to carry out these rules optimisations. The approach was also, developed and qualitatively evaluated in this chapter. Then, based on a weighted voting strategy, ReG-Rules builds a classification committee of rules for each classification attempt to decide the final ensemble prediction decision.

Three experimental studies were conducted in the this chapter, which can be summarised as follows:

1. ReG-Rules classifier was evaluated empirically and compared with the stand-alone G-Rules-IQR classifier using two types of evaluation methods (separate train and test datasets, and five-fold cross validation). It was found that both empirical evaluation approaches achieved similar results for all performance

metrics. ReG-Rules outperformed G-Rule-IQR on average and most cases. In fact, abstaining from classification, which is a typical problem of rule-based classifiers was almost non-existent in ReG-Rules. Hence, objective 4 (restated below) is met in this chapter:

> *"To develop an expressive ensemble learner footed upon the base classifier developed on objective 2 and the Rule Merging technique in objective 3."*

2. The ranking approach, which is integrated in ReG-Rules was examined empirically by implementing another version of ReG-Rules that does not involve any ranking process to its composite classifiers. It was found that ReG-Rules with ranking performed equal or better than the other version in all cases.

3. The Rule Merging (RM) algorithm, which is also integrated in ReG-Rules was evaluated qualitatively using two case studies, displaying the rule sets before and after merging, and it was found that merged rule sets are more compact and easier to read. Hence, objective 3 (restated below) is met in this chapter:

> *"To improve the quality of rule sets by developing rule merging techniques for predictive rules and minimising loss of accuracy.'*

Overall, it can be said that rule-based predictive models are among the most expressive classification techniques in data mining. Ensemble learners aim to improve classification performance, but generally, often at the expense of explainability. ReG-Rules successfully provides an approach to harvest the predictive power of an ensemble learner, while maintaining several explainable aspects of rule-based predictive models. However, further improvements to the results on objective 4 can be made. These proposed improvements will be introduced and empirically evaluated in the next chapter.

Chapter 6

CRC: a Significant Extension of the Ensemble ReG-Rules Learner

This chapter introduces a number of improvements that can be made to the ensemble ReG-Rules learner to increase its expressive power, and also to avoid some potential issues related to computations, and memory resources. These improvements resulted in a predictive ensemble learning system, called CRC (Consolidated Rules Construction), which can be considered a significant extension of the ensemble ReG-Rules learner. The chapters also, illustrates the new approach incorporated in CRC learner, termed 'CRC consolidator', which can be utilised by a rule-based ensemble models to merge multiple rule sets into a single global rule set.

6.1 Introduction

Several studies in the literature such as [2, 3, 7] argue that most ensemble classifiers are hardly readable by a human. However, in Chapter 5 a new predictive ensemble learner called ReG-Rules[1] was developed and explored it is possible to create an explainable ensemble model with rules that are comprehensible to a human user while benefiting from predictive performance of ensemble learning. Nevertheless, there are some improvements that can be made to the ReG-Rules ensemble system to avoid potential issues caused by combining the votes of many base classifiers in every classification attempt, which might be very expensive in terms of time and space.

[1] ReG-Rules, which is an explainable rule-based ensemble classifier, was one of the contributions from this project and published in [22].

Therefore, this chapter aims to further improve the results on objective 4 (restated below) by developing a new method of compressing a rule-based ensemble classifier into a single global model with a single consolidated rule set.

> *"To develop an expressive ensemble learner footed upon the base classifier developed on objective 2 and the Rule Merging technique in objective 3."*

This model is a significant extension version of the ensemble ReG-Rules system that can be more expressive while benefiting from predictive performance of ensemble learning in comparison with its stand-alone base learners. It is termed **'CRC'**, which is stand for **C**onsolidated **R**ules **C**onstruction and its general framework is expressed in the next section.

Section 6.2) introduces the consolidation approach, which can be utilised to merge multiple rule sets into a single global one in a way that can be used directly in the prediction stage. The approach is integrated in CRC learning system and termed 'CRC's Consolidator'. This new method is expressed in Algorithm 11 and is in line with objective 3:

> *"To improve the quality of rule sets by developing a rule merging technique for predictive rules and minimising loss of accuracy.'*

6.2 Framework for the Consolidated Rules Construction System: CRC

As mentioned in the previous section, the purpose of developing CRC ensemble learner is to increase the expressive power of the ensemble ReG-Rules learner while maintaining the key advantage of the ensemble systems, which is the high predictive accuracy comparing with the stand-alone classifiers. The general structure of CRC as shown in Figure 6.1 consists of five components: (1) Diversity Generation, (2) Base Classifier Inductions, (3) Models Selections, (4) Stacking and Consolidation, (5) Prediction. The explanations for each component are presented here by referring to the general framework (Figure 6.1) and to the lines of code in Algorithm 10.

Regarding the first three stages, please note that there are many similarities between CRC and ReG-Rules classifiers and therefore the following sections will only briefly describe these common stages, while further details are provided for stages 4 and 5, which describe the new CRC approach.

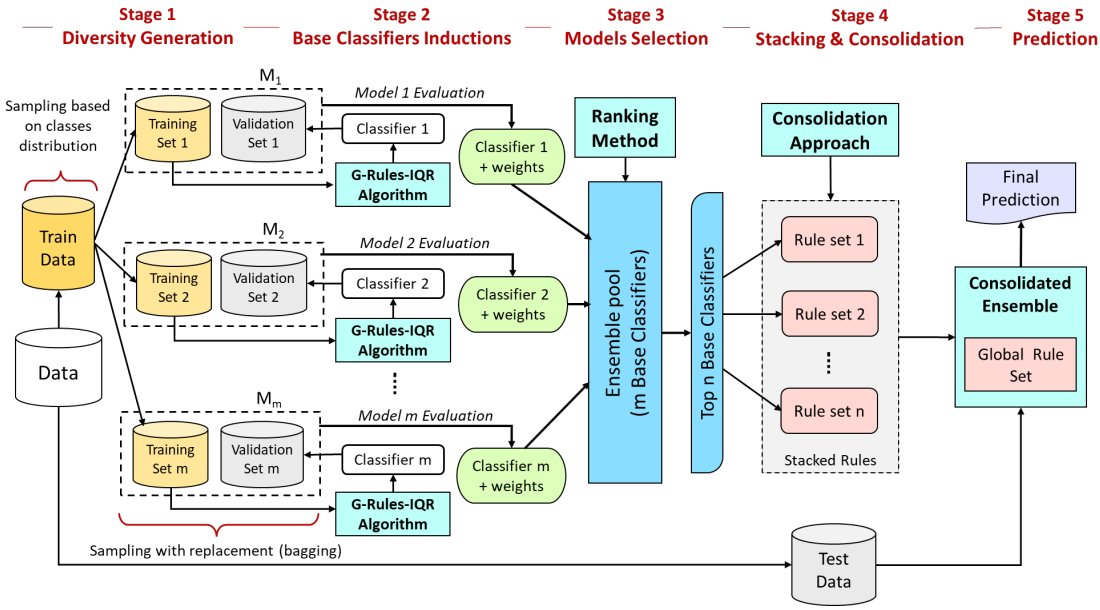


Figure 6.1: The General Framework of Consolidated Rules Construction (CRC) Classifier

6.2.1 Stage 1: Diversity Generation

The generation of a set of base classifiers should be as diverse as possible to assure producing uncorrelated errors and then obtain a more accurate ensemble [24, 73, 92]. The process of obtaining a diverse set of base classifiers in CRC system is very similar to that of ReG-Rules (see Section 5.3.1). It is based on introducing randomness into the data sampling technique and then building a group of different models from diverse training subsets. In other words, as shown in Figure 6.1 and Algorithm 10 (lines 1 to 3), CRC like in ReG-Rules uses the following sampling methods to encourage diversity between its base classifiers:

1. Sample a dataset randomly without replacement into train and test datasets. While the testing data is used only once to perform the general evaluation for the ensemble, training data will be used in the second sampling method.

2. Bagging, which is a sampling with replacement approach is used to produce multiple training and testing datasets from a single data source. In this method, the size of each training subset is equal to the original training data, i.e. statistically contains about 63.2% of instances. The remaining 36.8% of the instances that are not selected in the training phase, which called out-of-bag instances, can be used as a validation dataset to evaluate an individual classifier [9].

Algorithm 10: Consolidated Rules Construction: CRC

Notations: M: Number of models, S: Training dataset, V: Validation dataset, R: rule set, BC: base classifier, E_{pool}: Ensemble Pool

1 Randomly sample dataset without replacement into train and test datasets ($train$, $test$)
for $i = 1 \rightarrow M$ **do**
2 $s_i \leftarrow$ a random sample of $train$ dataset generated by Bagging method (sample with repalcement)
3 $v_i \leftarrow$ out-of-bag set
4 Generate a base classifier BC_i by applying Algorithm 5 (G-Rules-IQR) on s_i dataset and learn a rule set $\rightarrow R_i$
5 Evaluate BC_i performance by applying R_i on v_i dataset
6 Calculate a weight for each rule induced in previous line
7 Send BC_i including its rule set weights to the ensemble pool E_{pool}
8 **end**
9 Rank all the base classifiers BC collected in E_{pool} according to the criteria described in Section 5.3.3
10 Eliminate weak BC by selecting the n top models ($topBC$) ranked in the previous step according to the following if statement:
11 **if** *models selection type = defualt* **then**
12 $n \leftarrow 20\%$ M models
13 **else**
14 $n \leftarrow$ selected models size defined by user
15 Select the top n BC models in line 9
16 *SR* $\leftarrow$ stack all the rule sets induced by the n top models ($topBC$) in one large set
17 Apply Algorithm 11 (CRC Consolidator) to the stacked rules in *SR* set and produce a single consolidated rule set
18 Sort the individual rules in the consolidated set according to their quality
19 **return** *CRC Classifier*

6.2.2 Stage 2: Base Classifier Inductions

This stage is also very similar to that in ReG-Rules classifier (see Section 5.3.2). The number of base classifiers that should be generated to construct the ensemble model is specified simply as a predefined parameter that can be defined by user or by default value. This is referred to as M in Algorithm 10 and also in Figure 6.1. It can be seen in line 4 that CRC builds its M base classifiers in the same way as ReG-Rules does, i.e., it uses G-Rules-IQR (Algorithm 5, which was one of the contributions of this thesis and published in [21]) as the base inducer of these M base classifiers. Then, between lines 5 and 7, CRC performs the independent evaluation of each classifier using different validation dataset.

The evaluation method is called a 'classifier's performance weighting' and it was previously detailed in Section 5.3.2. Briefly, the weighting criteria of this method contains a combination of: (1) number of rules, (2) CUR: which is a track record prediction for each Correctly Used Rule during validation phase, (3) abstaining rate, (4) tentative accuracy. At the end of this stage, each base classifier will be associated with its weight and sent to the 'ensemble pool' as shown in Figure 6.1. This weight is reflecting the quality of a base classifier's rule set.

6.2.3 Stage 3: Models Selections

The number of base learners that should be included in the final model is an important factor for building an effective ensemble [24,114]. As previously explained in Section 5.3.3, by following the theorem of 'many could be better than all' [124], a smaller ensemble model can be extracted from a larger one without causing loss in its predictive performance. This also may enhance the comprehensibility of the ensemble. To achieve that, CRC uses a models selection approach, termed 'Ranking-based CUR', which is also used by ReG-Rules learner and it is previously detailed in Section 5.3.3. The process can be briefly described as follows:

1. Rank all the induced base classifiers according to their weights, which are derived from their classification performance on different validation datasets (Stage 2).

2. Select the top-ranked models according to a given threshold, which is a fixed user-specified amount or percentage of models (default = 20 %). Thus, the weak classifiers will be eliminated as highlighted in Algorithm 10 (lines 10 to 15).

6.2.4 Stage 4: Stacking and Consolidation

The previous stage ended by a group of top-ranked selected base classifiers; each have an individual rule set. Despite that these base models were constructed independently using diverse training subsets, their rule sets are still considered to be induced from the same source of data. Consequently, due to the 'theory of overlapping rules' that most separate and conquer approaches are based on, there is a possibility that some of these rules are overlapped. This may often result in smaller rule sets, which are less susceptible to the redundancy problem during the training phase [43]. However, overlapping rules are generally unnecessary as they need to be tested at prediction phase and thus incur unnecessary additional testing.

The ReG-Rules learning system has addressed this problem in Section 5.3.4 by locally applying a rule merging (RM) technique (Algorithm 8) among each individual rule set. Notably, this post-processing method of the induced rules has to be repeatedly applied to each base classifier in ReG-Rules. The results of the experiments conducted in Chapter 5 show that the RM approach is very effective in lowering the total number of rules. However, this section proposes an alternative method, called 'Rules' Stacking and Consolidation', that addresses the overlapping rules issue in which make the testing more efficient.

In this new method, CRC learner compresses the top base classifiers into a single global rule-based learner instead of locally merging each rule set independently. This is expected to enhance the expressiveness of the ensemble CRC learner to an extent that it is similar to a predictive standard base learner, thereby meeting the aim of this work. The approach as shown in Figure 6.1 consists of the following two steps:

1. *Stacking*: CRC classifier gathers all the rule sets of the top base classifiers and stacks them in one large set. This is represented by *'staked rules'* in Figure 6.1 and referred to as ' SR ' in Algorithm 10 (line 16). The key idea of stacking here is to simply accumulate the rule sets in the same order of their original ranked base classifiers, without performing any optimisation or filtering to the rules. Hence, the need to keep the base classifiers themselves no longer exists in which they can be simply discarded at this level.

2. *Consolidation*: CRC, in Algorithm 10 (line 17), integrates a method that will be detailed below to perform the global merging process and produces a single consolidated rule set.

Integrated Rules Consolidation Approach (CRC Consolidator)

After eliminating the base classifiers and stacking their rules in one large set, the decision about which rule will be kept, improved or even removed depends on the rule's quality (individual weight). The process as shown in Algorithm 11 (CRC consolidator) begins by initialising a new global rule set (line 1). Then each rule in the staked rule set (SR) is checked against the replications and the overlaps. If two rules (e.g. SR_1 , SR_2) are identical, one of them will be removed (line 6). Otherwise, SR_1 and SR_2 will be to considered as candidate overlapped rules. This is conditioned by the decision returned from Algorithm 7 (Overlap Checking),[2], which is invoked by the CRC Consolidator in line 8 to carry out the examination. A decision (true/false) about the current rules examination is retuned to the CRC consolidator.

Next, the CRC Consolidator continues at line 10 where the current overlapped rules are considered for the final consolidation process and a new iteration will be started again to examine another two new rules until all the rules in the staked set (SR) are investigated. Then, a process of constructing a new consolidated rule from a number of overlapped rules is started in line 14. First, the overlapped rules are grouped by terms. Then, depending on the type of attribute in each term, the

[2] For detailed description of Algorithm 7 (Overlap Checking), the reader is referred to Chapter 5 (Section 5.3.4)

process will continue. In case of categorical attributes, a new term is generated in the form $(\alpha = v)$ where α is the attribute name and v is a discrete value that occurs in all the current overlapped terms. If the attribute type is continuous, a new term is generated in the form $(x < \alpha \leq y)$ where x is the smallest lower bound presented in all the current overlapped terms and y is the largest upper bound presented in the same overlapped terms. After the term is created, it will be appended to the new consolidated rule (line 24). Then, a new iteration of the next term will be started. Finally, in line 26, all the consolidated rules are added to the global rule set. The weight of each consolidated rule is estimated by averaging the weights associated to all the overlapped rules used in its generation.

Algorithm 11: Consolidation Approach: CRC Consolidator

```
1  Initialise new GlobalRules set
2  for (i = 1 → SR ) do
3      OverlappedRules ← SR_i ;
4      for (j = 1 → SR [ − OverlappedRules] ) do
5          if (SR_i and SR_j are identical rules) then
6              Skip current SR_j
7          else
8              OverlapExist ← Apply Algorithm 7 (Overlaps Checking) on SR_i
                and SR_j
9              if (OverlapExist = True ) then
10                 OverlappedRules ← ADD (SR_j)
11             end
12         end
13     end
14     if (OveralppedRules list contains rules other than SR_i) then
15         ConsoR ← empty        // a new consolidated rule intialisation
16         foreach ( α in OveralppedRules list) do
17             if (attribute α is categorical) then
18                 Create a rule-term α_j in the form (α = v) ;
19             else if (attribute α is continuous) then
20                 x ← smallest lower bound of α ;
21                 y ← largest upper bound of α ;
22                 Create a rule-term in a fom of (x < α ≤ y)
23             end
24             Append a rule-term built in lines 18 or 22 to the new consolidated rule
                ConsoR
25         end
26         GlobalRules set ← ADD (ConsoR)
27     end
28 end
29 return new GlobalRules set
```

6.2.5 Stage 5: Prediction

Combining multiple individual models' predictions (votes) promises a considerable increase in predictive accuracy compared with a single classifier. However, it has a number of potential issues in training and testing phases, as mentioned in Section 6.1. Issues related to the prediction stage can be summarised as follow:

- The number of base classifiers might be a bottleneck for the overall prediction time regardless of the voting method employed to produce the final ensemble decision.
- Most importantly, the resulting prediction may also be harder to explain or to justify by a human, which is particularly crucial for rule-based classifiers.

Most predictive ensemble learning systems including ReG-Rules may encounter the aforementioned obstacles during the prediction tasks. However, ReG-Rules alleviates these problems by reducing the number of base classifiers that should be combined to decide the final predictions (see Section 5.3.5). Ideally, the amount of reduction is determined by the user; however, in the experiments conducted in Chapter 5, ReG-Rules successfully achieved a 80% reduction in the ensemble size and offered a greater level of explainability without sacrificing accuracy.

Nevertheless, for each new unlabelled instance, ReG-Rules still needs to combine several or all the base classifiers' predictions and then build a committee of rules to decide the final prediction using a weighted voting strategy. In other words, the expressive power of ReG-Rules may rely on the size of the classification committees and the complexity of the datasets. Therefore, in order to assure a high level of expressiveness in the ensemble predictions, classifying a new instance should be done the same way as in single rule-based classifiers using a single rule set. The empirical evaluation, which is presented in the next section shows that this is successfully provided in stage 4 of CRC learning system using the global consolidated rule set.

6.3 Empirical Evaluation of CRC Learning Model

The aim of the experimental evaluation in this chapter is to evaluate the performance of the proposed ensemble CRC classifier comparing with: (1) the ensemble ReG-Rules model, which is developed in this research in Chapter 5 and published in [22], (2) G-Rules-IQR, the stand-alone classifier that has been chosen to be the base learning algorithm in the ensembles [3].

[3] G-Rules-IQR is also one of the contributions of this research described in Chapter 4 and published in [21]

6.3.1 Experimental Setup

All the experiments were performed on a 2.9 GHz Quad-Core Intel Core i7 machine with memory 16 GB 2133 MHz LPDDR3, running macOS Big Sur version 11.4. All the 24 datasets used in the experiments were chosen randomly from the UCI repository [17], the only conditions being that they contain continuous attributes and involve classification tasks. The specifications of the datasets are highlighted in Table 6.1 in terms of number of instances, attributes (including the type of attribute) and class labels. Datasets 15, 16 and 24 included few missing values in continuous attributes. To address this issue, the current research adopted and implemented a common method, found in the literature [5], which is based on estimating a missing numeric value with the average value for the concerning attribute.

Table 6.1: Characteristics of the datasets used in the experiments in Chapter 6

No.	Dataset	No. Attributes	No. Classes	No. Instances
1.	iris	5 (4 cont)	3	150
2.	seeds	8 (7 cont)	3	210
3.	wine	14 (13 cont)	3	178
4.	blood transfusion	6 (5 cont)	2	748
5.	banknote	6 (5 cont)	2	1,372
6.	ecoli	9 (7 cont, 1 name)	8	336
7.	yeast	10 (8 cont, 1 name)	10	1,484
8.	page blocks	11 (10 cont)	5	5,473
9.	user modelling	6 (5 cont)	4	403
10.	breast tissue	11 (10 cont)	6	106
11.	glass	11 (10 cont, 1 id)	7	214
12.	HTRU2	10 (9 cont)	2	17,898
13.	magic gamma	12 (11 cont)	2	19,020
14.	wine quality-white	13 (12 cont)	11	4,898
15.	breast cancer	12 (10 cont, 1 id)	2	699
16.	post operative	10 (1 cont, 9 categ)	3	90
17.	wifi localization	8 (7 cont)	4	2,000
18.	indian liver patient	12 (10 cont, 1 categ)	2	583
19.	sonar	62 (61 cont)	2	208
20.	leaf	17 (15 cont, 1 name)	40	340
21.	internet firewall	12 (cont)	4	65,532
22.	bank marketing	17 (6 cont, 10 categ)	2	45,211
23.	avila	11 (10 cont)	12	20,867
24.	shuttle	10 (9 cont)	7	58,000

The two ensembles (ReG-Rules and CRC) and their stand-alone inducer (G-Rule-IQR) have been implemented in the statistical programming language R [147] [4].

[4]All the source codes are available in a public online repository at `https://github.com/ManalAlmutairi/PhD_Project_Codes/tree/v1.0.0` and are archived at `https://doi.org/10.5281/zenodo.5557590` [23].

The source code used to implement CRC algorithm is similar to that for ReG-Rules differing only in the methodological aspects described in stage 4 (Sections 6.2.4), and stage 5 (Section 6.2.5). All the algorithms are evaluated against 5 metrics for classifiers, which are presented below:

1. Number of Rules: this is to compare the average number of rules generated by ReG-Rules' base classifiers, the average number of rules generated by CRC's base classifiers, and the total number of rules induced by the stand-alone G-Rules-IQR classifier. A low number of rules is desired.

2. F1 score: this is also known as the harmonic mean of precision and recall. This is a number between 0 and 1. A high F1 score is desired.

3. Accuracy: this is the ratio of instances that have been correctly classified either using the induced rule set or the majority class strategy in case of unclassified instances. This is a number between 0 and 1.

4. Tentative Accuracy: this is the ratio of instances that have been correctly classified using only the induced rule set, i.e. does not count the ones that are not covered by rules.

5. Abstaining Rate: this is the proportion of cases a classifier abstains from classification, i.e. the proportion of instances not covered by the rule set. This is a number between 0 and 1. A low abstaining rate is desired.

6. Execution Time: this is the time needed to complete all the stages and produce the final decisions.

Please consider that there is a relationship between accuracy, tentative accuracy and abstaining rate. Tentative accuracy does not consider the abstained instances, while the accuracy counts them as misclassifications. Hence, the higher the abstaining rate, the higher the tentative accuracy and the lower the accuracy.

The methodology used for experimentation with the 24 datasets is hold-out procedure; each dataset was randomly sampled without replacement into train and test datasets. While the 70% of the data instances were used to train and build the ensemble classifier, the remaining 30% were used as a testing dataset. In case of the ensemble models (ReG-Rules and CRC), the training dataset is used to generate multiple base classifiers using bagging, which is a method of sampling with replacement that have been used in this work to ensure a sufficient level of diversity during the ensemble construction whereas the test set is used only once to assess the general performance of the classification models.

6.3.2 Results and Discussion

This section presents the experimental results of CRC ensemble learning model in comparison with: (1) ReG-Rules ensemble learner and (2) the stand-alone G-Rules-IQR, which is also the inducer of the base classifiers in both ensembles. In each table, the # symbol refers to the number of the dataset in Table 6.1. The best result(s) in the tables for each dataset are highlighted in bold letters. The tables show the results with respect to the 6 evaluation metrics described in the previous section. With respect to F1 score, overall accuracy, tentative accuracy and learning time, Table 6.3 shows the comparison between CRC and ReG-Rules while Table 6.4 presents the comparison between CRC and G-Rules-IQR in these metrics, which will be discussed .

Regarding the 'number of induced rules' and the 'abstaining rates' metrics, Table 6.2 shows the results of all the three classifiers. However, considering that G-Rules-IQR is a stand-alone classifier; it is not reasonable to be compared with the ensemble classifiers with respect to 'the number of rules' measurement. Therefore, CRC learner is only compared with ReG-Rules ensemble. As can be seen in the table, the number of consolidated rules produced by CRC is considerably smaller than the total number of rules produced by ReG-Rules in all datasets, making it expressive and much easier for the analyst to understand for prediction. In most cases, the size of the rules generated by CRC is reduced by 90% compared with ReG-Rules' sizes as shown in Figure 6.2. Abstaining from classification, a typical problem of rule-based classifiers, was almost non-existent in both ensembles (ReG-Rules and CRC) compared with the stand-alone G-Rules-IQR's abstaining rates, which were higher by more than 10% on several datasets compared with ReG-Rules and CRC. In four datasets (9, 10, 18 and 20) the abstaining rate in G-Rules-IQR reaches 30%, 19% , 18% and 40% respectively.

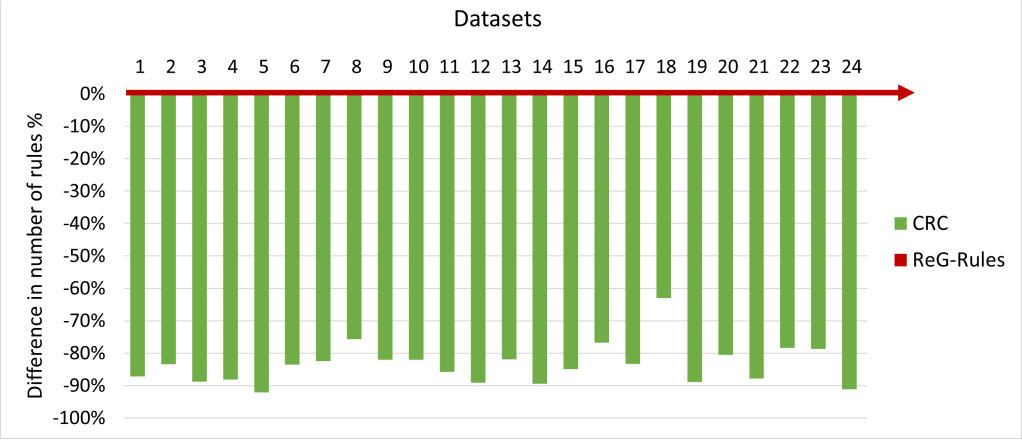


Figure 6.2: Differences in percentage of number of rules generated by CRC compared with ReG-Rules

Table 6.2: Number of Rules and Abstaining Rates for CRC learner compared with ReG-Rules and G-Rules-IQR learners

#	Number of Rules			Abstaining Rate		
	G-Rules-IQR	ReG-Rules	CRC	G-Rules-IQR	ReG-Rules	CRC
1	*18*	342	**44**	0.07	**0.00**	**0.00**
2	*22*	386	**64**	0.03	**0.00**	**0.00**
3	*13*	250	**28**	0.06	**0.00**	**0.00**
4	*20*	321	**38**	**0.00**	**0.00**	**0.00**
5	*89*	1630	**128**	0.02	**0.00**	**0.00**
6	*37*	649	**107**	0.02	**0.00**	**0.00**
7	*99*	1648	**289**	0.04	**0.00**	**0.00**
8	*131*	2348	**570**	0.02	**0.00**	**0.00**
9	*57*	901	**162**	0.30	**0.00**	0.01
10	*28*	485	**87**	0.19	**0.00**	**0.00**
11	*30*	505	**72**	0.11	**0.02**	**0.02**
12	*35*	521	**57**	**0.00**	**0.00**	**0.00**
13	*79*	1388	**251**	**0.00**	**0.00**	**0.00**
14	*126*	2289	**243**	0.02	**0.00**	**0.00**
15	*11*	186	**28**	**0.00**	**0.00**	**0.00**
16	*29*	451	**105**	0.11	**0.00**	**0.00**
17	*59*	955	**159**	0.01	**0.00**	**0.00**
18	*47*	996	**368**	0.17	**0.00**	**0.00**
19	*16*	270	**30**	0.13	**0.00**	**0.00**
20	*124*	2015	**393**	0.40	**0.01**	0.03
21	*21*	402	**49**	**0.00**	**0.00**	**0.00**
22	*115*	1977	**428**	0.01	**0.00**	**0.00**
23	*158*	3272	**699**	0.09	**0.01**	**0.01**
24	*27*	428	**38**	**0.00**	**0.00**	**0.00**

Comparing with ReG-Rules Ensemble Learner:

Table 6.3 shows the comparison of the performance of CRC and ReG-Rules using F1 score, overall accuracy, tentative accuracy and learning time. The results of F1 score reveals that CRC performs equal or better than ReG-Rules in 13 out of 24 datasets, as can be seen in Figure 6.3. Also, CRC was very competitive on 4 out of the remaining datasets in which it was only underperformed by a maximum difference of 3%.

Please note that the comparison between CRC and ReG-Rules in terms of overall accuracy and tentative accuracy are very similar[5]. The results show that with respect to both metrics, CRC performs at the same level as ReG-Rules in 14 out of 24 cases. On 5 out of the remaining 10 datasets (2, 6, 11, 18 and 19) where CRC did not achieve the highest accuracy and tentative accuracy, it was still very close within 3% of the best results compared with ReG-Rules learner. On one dataset (#5), the accuracy and tentative accuracy of CRC were lower than ReG-Rules by about 15%. However, this is also the dataset where the highest compression in the size of the rules is taking place.

[5]This is related to the nature of relationship between accuracy and tentative accuracy, which is based on the abstaining rates, i.e. almost non-existent in CRC and ReG-Rules as Table 6.2 shows.

Table 6.3: Comparison of the performance of CRC and ReG-Rules using F1 score, Overall Accuracy, Tentative Accuracy and Learning Time

#	F1 score		Accuracy		Tentative Accuracy		Learning Time (Sec.)	
	ReG-Rules	CRC	ReG-Rules	CRC	ReG-Rules	CRC	ReG-Rules	CRC
1	**0.93**	**0.93**	**0.93**	**0.93**	**0.93**	**0.93**	112.8	**94.8**
2	**1.00**	0.97	**1.00**	0.97	**1.00**	0.97	232.2	**192.6**
3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	166.2	**148.8**
4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	358.8	**322.2**
5	**0.99**	0.87	**0.99**	0.84	**0.99**	0.84	3251.4	**2939.4**
6	**0.91**	**0.91**	**0.95**	0.92	**0.95**	0.92	384	**360**
7	**0.91**	0.83	**0.97**	**0.97**	**0.97**	**0.97**	3262.8	**3096**
8	**0.87**	0.82	**0.98**	**0.98**	**0.98**	**0.98**	10512	**9612**
9	**0.95**	0.87	**0.94**	0.86	**0.94**	0.87	733.2	**631.2**
10	**0.97**	0.91	**0.97**	0.91	**0.97**	0.91	245.4	**228.6**
11	**0.93**	0.90	**0.95**	0.94	**0.97**	0.95	264	**247.2**
12	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	12600	**12168**
13	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	18288	**17100**
14	0.87	**0.99**	**1.00**	**1.00**	**1.00**	**1.00**	12240	**11880**
15	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	174.6	**156**
16	**0.77**	**0.77**	**0.63**	**0.63**	**0.63**	**0.63**	193.2	**189**
17	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	2833.2	**2635.8**
18	**0.84**	0.82	**0.73**	0.71	**0.73**	0.71	2095.8	**1951.8**
19	**0.97**	0.96	**0.97**	0.95	**0.97**	0.95	897.6	**864.6**
20	**0.67**	0.61	**0.56**	0.46	**0.57**	0.47	2388	**2125.8**
21	**0.90**	**0.90**	**1.00**	**1.00**	**1.00**	**1.00**	71316	**39276**
22	**0.99**	**0.99**	**0.98**	**0.98**	**0.98**	**0.98**	41400	**32940**
23	**0.93**	0.86	**0.92**	0.86	**0.92**	0.86	119232	**68292**
24	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	20808	**17964**

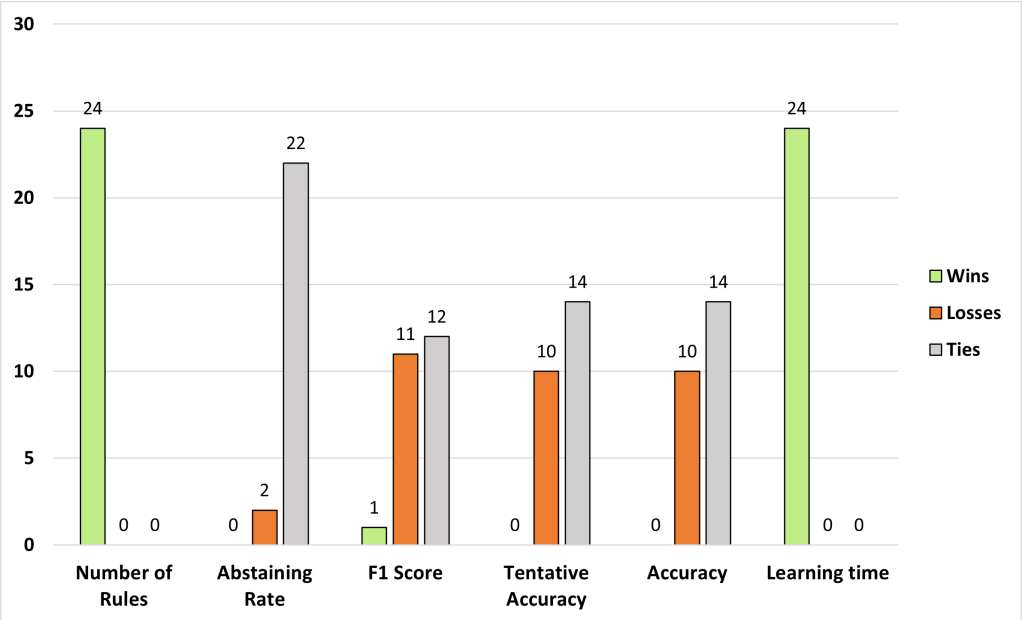


Figure 6.3: Performance of CRC approach compared with the ensemble ReG-Rules approach

Regarding the learning time, Table 6.3 demonstrates that CRC learner is faster than ReG-Rules on all datasets. The decrease in learning times was up to 45% in some cases as shown in Figure 6.4.

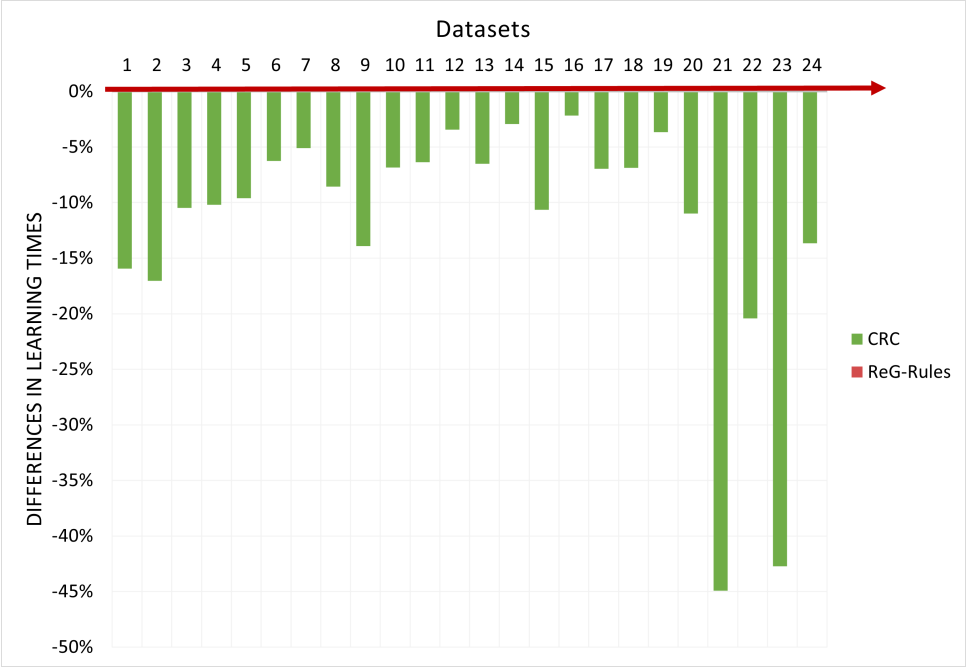


Figure 6.4: Differences in percentage of learning times of CRC compared with ReG-Rules

Comparing with Stand-alone G-Rules-IQR Learner

On the other hand, the performance of CRC learning model is also compared with its stand-alone inducer (G-Rules-IQR algorithm). Table 6.4 shows the results of this comparison using F1 score, accuracy and tentative accuracy. CRC achieves the best F1 scores in 14 out of 24 datasets, as can be seen in Figure 6.5. In the cases where CRC did not outperform G-Rules-IQR, its scores only marginally lower. For example, the differences in 5 cases (1, 2, 7, 18 and 23) were less than 4%.

In terms of overall accuracy, as can be seen in Table 6.4 and Figure 6.5, CRC achieves the highest results in most cases (21 out of the 24 datasets). Moreover, CRC achieves the highest tentative accuracies in 14 out of 24 datasets compared with G-Rules-IQR classifier. CRC was also very competitive with G-Rules-IQR, in 7 out of the remaining 8 datasets their results are very close. Only on one dataset (# 20), CRC's tentative accuracy was much lower than the stand-alone G-Rules-IOR. However, this dataset also causes the highest abstaining rate for G-Rules-IQR in the current experiments, and therefore it had been classified using the majority class label method. As mentioned in Section 6.3.1 the abstained instances are not considered in the tentative accuracy.

Table 6.4: Comparison of the performance of CRC and G-Rules-IQR using F1 score, Overall Accuracy and Tentative Accuracy

#	F1 score		Accuracy		Tentative Accuracy	
	G-Rules-IQR	CRC	G-Rules-IQR	CRC	G-Rules-IQR	CRC
1	**0.96**	0.93	0.91	**0.93**	**0.95**	0.93
2	**1.00**	0.97	**0.97**	**0.97**	**1.00**	0.97
3	0.98	**1.00**	0.94	**1.00**	0.98	**1.00**
4	0.98	**1.00**	0.97	**1.00**	0.97	**1.00**
5	**0.98**	0.87	**0.98**	0.84	**0.99**	0.84
6	**0.99**	0.91	**0.96**	0.92	**0.97**	0.92
7	**0.87**	0.83	0.93	**0.97**	0.96	**0.97**
8	**0.93**	0.82	**0.98**	**0.98**	**0.99**	0.98
9	**0.95**	0.87	0.72	**0.86**	**0.95**	0.87
10	0.77	**0.91**	0.66	**0.91**	0.81	**0.91**
11	0.86	**0.90**	0.86	**0.94**	**0.97**	0.95
12	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
13	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
14	0.96	**0.99**	0.97	**1.00**	0.99	**1.00**
15	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
16	0.49	**0.77**	**0.67**	0.63	**0.67**	0.63
17	**1.00**	**1.00**	0.99	**1.00**	**1.00**	**1.00**
18	**0.83**	0.82	**0.71**	**0.71**	**0.71**	**0.71**
19	0.95	**0.96**	0.87	**0.95**	0.94	**0.95**
20	**0.75**	0.61	0.39	**0.46**	**0.65**	0.47
21	**0.90**	**0.90**	**1.00**	**1.00**	**1.00**	**1.00**
22	**0.99**	**0.99**	**0.98**	**0.98**	**0.98**	**0.98**
23	**0.89**	0.86	0.85	**0.86**	**0.88**	0.86
24	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

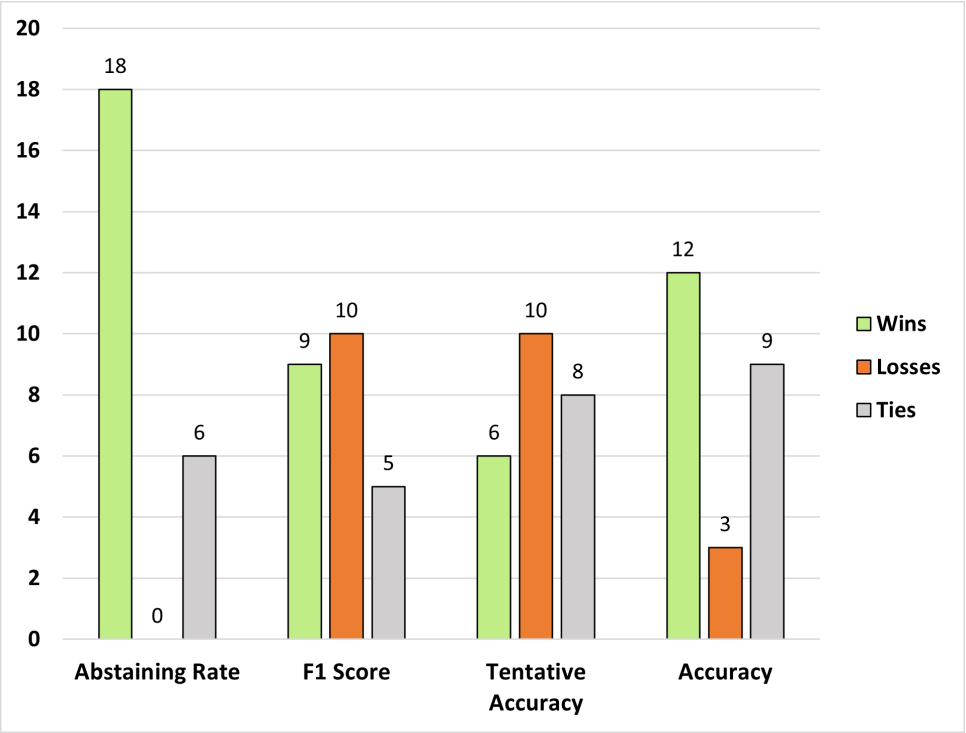


Figure 6.5: Performance of CRC approach over the stand-alone G-Rules-IQR approach

6.4 Summary

The purpose of this chapter was to increase the expressive power of rule-based ensemble learning models while maintaining the key advantage of the ensemble learners, which is the high predictive accuracy compared with the stand-alone classifiers. Therefore, a new algorithm was developed in Section 6.2 to compress the ensemble ReG-Rules learner into a single global classifier, which can be used directly in predictions without the need to combine multiple classifiers' votes on every classification attempt. The new proposed ensemble learner was called Consolidated Rules Construction (CRC) and consists of 5 components: Diversity Generation, Base Classifiers Induction, Models Selection, Stacking and Consolidation and Prediction.

CRC induces a diverse ensemble based on bagging, just like in ReG-Rules (see Section 5.3.1). The induced base models are ranked according to their classification performance on validation datasets, and only the best performing models are retained and considered for the global rules' consolidation process. To measure the individual classification performance of these base models, CRC uses a weighting method composed of various metrics (see Section 6.2.2). Moreover, to rank and then select the best performing models, CRC utilises a ranking-based method (see 5.3.3). The rule sets of these best ranked classifiers are consolidated in a single global rule set using a new approach called 'Rules Stacking and Consolidation', which was integrated in CRC (see Section 6.2.4).

CRC was empirically evaluated in Section 6.3 and compared with the ensemble ReG-Rules classifier and the stand-alone G-Rules-IQR classifier. The results of these comparisons can be summarised as follows:

- Compared with the ensemble ReG-Rules classifier, CRC considerably outperformed in terms of number of rules that have been constructed and used for predictions in all cases. Figure 6.2 shows this significant difference between the two ensembles, which reaches 90% in most cases. Abstaining from classification, a typical problem of rule-based classifier, was almost non-existent in both ensembles. Also Figure 6.3 reveals that CRC was competitive compared with ReG-Rules in terms of F1 score, overall accuracy and tentative accuracy. Regarding the learning time metric, CRC outperformed ReG-Rules in all cases. Figure 6.4 shows that the decreases in CRC's learning times reached the 45% in some case compared with ReG-Rules.

- Compared with the stand-alone G-Rules-IQR classifier, CRC outperformed in terms of abstaining rates in all cases. Also, CRC achieved the highest results on most cases in terms of F1 score, overall accuracy and tentative accuracy.

Generally speaking, developing the new consolidation approach (CRC Consolidator) in this chapter, which was utilised by CRC learner to merge multiple rule sets into an expressive global rule set, is in line with the objective 3:

> *"To improve the quality of rule sets by developing a rule merging technique for predictive rules and minimising loss of accuracy.'*

The global rule set constructed by the CRC learner can assure a high level of expressiveness in the ensemble predictions, just the same way as in the single rule-based classifiers. Therefore, this chapter has successfully fulfilled the research objective 4:

> *"To develop an expressive ensemble learner footed upon the base classifier developed on objective 2 and the Rule Merging technique in objective 3."*

Chapter 7

Conclusion and Future Directions

In this concluding chapter, a summary of the research work described in this thesis is provided in Section 7.1. Section 7.2 presents the contributions to knowledge and the extent to which the project aim and objectives have been met is examined. Also, it lists the publications that have been produced during the project. The chapter then, in Section 7.3 describes some potential areas for future research, by which this work could be extended.

7.1 Summary of Thesis

In many critical applications, such as medical diagnoses, financial analysis, credit risk evaluation, terrorism detection, etc. Predictive learning models are required to be not only reliable and accurate, but also comprehensible to avoid or reduce the risk of irreversible misclassification. Constructing a single classification model may lead to overfitting, a common problem in data mining that causes a model to perform badly on testing data. Often, the standard strategy in data mining to reduce overfitting involves sacrificing accuracy on the training data for accuracy of classifying unseen data. However, this trading-off strategy may not be tolerated by a human especially in critical applications. Alternatively, ensemble learning has initially been developed in order to address overfitting, while improving (not only maintaining) the predictive accuracy of the learner. However, this often goes at the expense of expressiveness and explainability of the predictive model learned, as the human analyst is presented with a large number of different classification models. This provided the motivation for the research presented in this thesis, which aims to develop a predictive learner that can be expressive while retaining key advantages of ensembles learners.

In **Chapter 2**, the literature around predictive learning systems has been categorised into two paradigms: ensemble learning system and stand-alonelearning

system. With respect to the ensemble paradigm, the chapter has illustrated the general concept and the philosophy of ensemble learning systems. Then a number of ensemble methods characteristics have been thoroughly explored, followed by describing a number of criteria for evaluating the predictive performance of ensembles. Regarding stand-aloneclassifiers, the chapter has reviewed a number of predictive data mining algorithms, which involved investigating their expressiveness and the extent of being a suitable base learning algorithm for the proposed ensemble. Modular rule induction has shown to be superior in terms of the expressiveness level compared with other classification algorithms. Therefore a choice was made to develop a rule-based algorithm in relation to the Prism family of algorithms to be used as the base learners of the ensemble.

The rationale behind this choice was discussed in **Chapter 3**. The main advantages of Prism family of algorithms have been identified, followed by analysing a number of practical and computational issues found in existing versions of this algorithm family. Also, the chapter has reviewed the solutions, which are currently being deployed to overcome these issues. In particular, the focus was on approaches to deal with continuous features, which is based on converting them into categorical features at each stage of the induction process by using frequent cut-point calculations method (also referred to as binary splitting). A numeric rule-term that can be produced using this type of local discretisation method is in the form ($\alpha < v$ and $\alpha \geq v$) which is computationally expensive. This problem was thoroughly investigated in Chapter 3 besides reviewing a number of common (local and global) discretisation techniques. Then a computationally more efficient and expressive approach to induce numeric rule-terms in the form ($x \leq \alpha < y$) has been proposed. This new rule-term structure, which is based on Gaussian Probability Density Distribution (*GPDD*) greatly enhances the readability of the individual rules. Therefore, it has been utilised to develop two new members in the Prism family of algorithms, namely, *'G-Prism-FB'* and *'G-Prism-DB'*. The letter 'G' stands for Gaussian, 'FB' refers to fixed size rule-term boundaries, and 'DB' stands for dynamic size rule-term boundaries. An empirical performances evaluation of G-Prism algorithms have been detailed in Chapter 3 which concluded that (in general) G-Prism-DB performs better than G-Prism-FB and than the original Prism algorithm in most cases. However, it is possible that this approach could be improved further.

Therefore, **Chapter 4** started by highlighting the shortcomings that exist in G-Prism-FB and G-Prism-DB algorithms, which are: (1) the more accurate G-Prism-DB requires a user defined rule-term boundary threshold, (2) both algorithms assume normally distributed continuous attributes, (3) both algorithms have a higher abstaining rate than the basic Prism classifier and (4) G-Prism-DB was es-

timated to have a longer execution time than G-Prism-FB and the basic Prism. Then the new proposed solutions were deployed to deal with the aforementioned limitations, resulting in a new rule induction classifier termed *'G-Rules-IQR'*. The algorithm is based on a combination of Gauss Probability Density Distribution (GPDD), InterQuartile Range (IQR) and data transformation approach towards normally distributed data. For comparative purposes, three variations of original Prism with different discretisation methods (binary splitting, ChiMerge and Caim) have been implemented. Also, the implementations of G-Rules-IQR and G-Prism algorithms have allowed to switch off the transformation to approximate normal distribution.[1] The aforementioned (five) approaches have been empirically evaluated and compared with each other using six metrics: number of rules, abstaining rates, F1 score, accuracy, tentative accuracy and execution time. With respect to number of rules induced, G-Rules-IQR has produced fewer rules compared with G-Prism algorithms but more rules compared with the original Prism. A small number of rules is desired. This is the only metric where the original Prism outperforms G-Rules-IQR classifier and G-Prism classifiers. Also, despite that G-Rules-IQR suffers from high abstaining rates in some cases; the comparisons have shown that there is no clear winner with respect to this particular metric. With respect to the rest of the remaining metrics, the results revealed that G-Rules-IQR with transformation outperformed its competitors in most cases.

Hence, **Chapter 5**, has presented this improved version of the G-Rules-IQR as the suitable base inducer of the proposed ensemble classifier, which is termed *'ReG-Rules'*. This choice was because G-Rules-IQR induces a highly expressive rule set and provides a high classification accuracy in comparison with other rule-based learners. The purpose of the ensemble ReG-Rules learner is to maximise the overall accuracy, while maintaining a high level of explainability in terms of rule examinations needed for tracing individual predictions. The general framework of ReG-Rules model has been demonstrated. It consists of 5 stages: Diversity Generation, Base Classifiers Induction, Models Selection, Rules Improvement, and Combination and Prediction. First, ReG-Rules uses bagging to ensure generating a diverse set of base classifiers (stage 1). Then a new ranking method is developed in ReG-Rules to rank the base models induced according to their classification performance on different validation datasets (stage 2). Only the best performing models will be selected and considered for predicting class labels, whereas the weak classifiers will be eliminated (stage 3). Next, a new local rule merging method is developed in ReG-Rules to further optimise the rule sets of its top-ranked models

[1] All the source codes are available in a public online repository at `https://github.com/ManalAlmutairi/PhD_Project_Codes/tree/v1.0.0` and are archived at `https://doi.org/10.5281/zenodo.5557590` [23].

by merging overlapping rules within each rule set independently (stage 4). Last, to avoid a potential problem of reliability between the top base models predictions, a new combination technique based on a weighted voting strategy, is developed in ReG-Rules to build a classification committee for each unseen instance and thus determine the final ensemble predictions (stage 5). Several experimental studies are conducted in Chapter 5 to evaluate empirically and qualitatively the general performance of ReG-Rules system and the new methods integrated in it. The results have shown that ReG-Rules has produced fewer rules per base classifier compared with its stand-alone G-Rules-IQR. Also, it was almost never abstained from classification compared with G-Rules-IQR, which has suffered from high abstaining rate on several datasets. In terms of F1 score, accuracy and tentative accuracy, ReG-Rules has outperformed G-Rules-IQR in most cases. Overall, the chapter has concluded that ReG-Rules successfully provides an approach to harvest the predictive power of an ensemble learner, while maintaining several explainable aspects of rule-based predictive models.

Nevertheless, there are some improvements that can be made to avoid some potential challenges related to expressiveness, computations and memory resources, especially when dealing with large datasets. These improvements that have been presented in **Chapter 6** are resulted in a significant extension of the ensemble ReG-Rules learner, which is called *'Consolidated Rules Construction (CRC)'* system. The chapter was aimed to further improve the results on objective 4 (restated in the next section), by developing a new consolidation approach to compress a rule-based ensemble learner into a single global classifier. CRC learner assures a high level of explainability in the ensemble predictions just the same way as in the single rule-based classifiers while preserving the key advantage of the ensemble, which is the high predictive accuracy compared with the stand-alone classifiers. The general framework of CRC has been demonstrated, and it consists of 5 components: Diversity Generation, Base Classifier Inductions, Models Selection, Stacking and Consolidation, and Prediction. There have been some similarities between CRC and ReG-Rules in the first three stages. However, in stage 4, the new consolidation approach is integrated in CRC to compress all the top selected base classifiers into a single global classifier, which produces a consolidated rule set that can be used directly in predictions without the need to combine multiple votes from multiple classifiers on every classification attempt like in ReG-Rules (stage 5). CRC was empirically evaluated through two different comparisons. First, comparing with the ensemble ReG-Rules, the results have shown that CRC was faster and more expressive in all cases. In fact, the number of rules constructed by CRC have been significantly decreased by 90% in most cases compared with ReG-Rules. In terms of F1 score, overall accuracy and tentative accuracy, CRC has

not outperformed ReG-Rules in all cases, but it was very competitive. Regarding the second comparison, which was with the stand-alone G-Rules-IQR classifier, the results revealed that the abstaining from classification was almost non-existent in CRC while it was higher than 10% in several cases in G-Rules-IQR. Also, with respect to the remaining metrics (F1 score, overall accuracy, tentative accuracy), CRC learner has performed better in most cases.

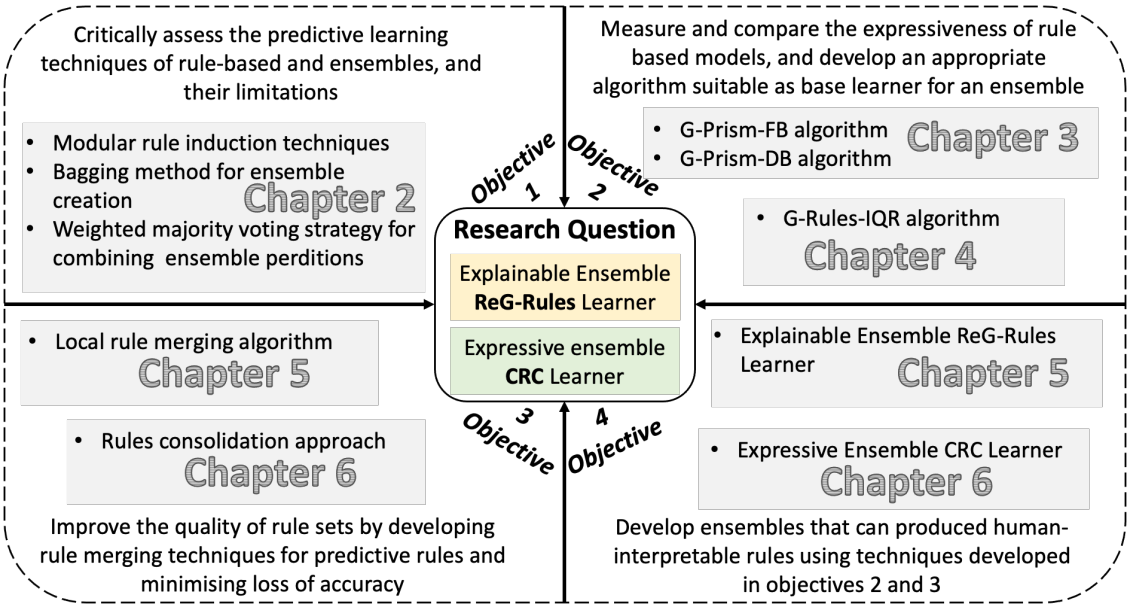


Figure 7.1: Overview of how research objectives are met to answer the research question

7.2 Research Findings and Contributions

This section will identify the main findings from the research work presented in this thesis, beginning with how this work has met the stated objectives highlighted in Chapter 1 in order to answer the research question. This is followed by a discussion of some key insights. Last, the contributions to knowledge made by this PhD project will be described.

7.2.1 Research Findings

To begin with, the main research question to be answered was:

Is it possible to develop a predictive ensemble model, which exhibits a similar expressiveness as the predictive base learner while improving its accuracy and lowering its abstaining rate?

In order to answer this research question, as can be seen in Figure 7.1 four objectives were required to be met. The work described in the thesis meets each of these research objectives as follows:

1. **To critically assess the predictive learning techniques of rule-based and ensembles and their limitations.**
   Various machine learning algorithms exist to induce classification models that can be used as base classifiers for a predictive ensemble learner. Therefore, the essential background in predictive data mining algorithms have been examined in this thesis to find the most appropriate techniques to be employed to answer the above research question. However, it was of interest in this study to focus on approaches that share the common goal of producing models that can be read by a human. Thus, black box approaches such as SVM, Kernel-based, ANN, deep learning, etc. have been excluded. Also, tree-based models have been found not sufficiently expressive, as they tend to be complex when they reach a certain size. On the other hand, it has been found that modular rule induction approaches are much closer to the white box models than other techniques.

2. **To measure and compare the expressiveness of rule based models and develop an appropriate rule-based predictive algorithm suitable as base learner for an ensemble.**
   Modular rule induction algorithms offer a greater explainability about how they arrive at a particular prediction compared with decision trees. Therefore, a choice was made to use a rule induction approach to produce a human-readable (expressive) rule set required by the objectives. The research has investigated the ability to learn from continuous attributes in a separate and conquer rule-based algorithm. In particular, the computational efficiency and the expressive power of rules have been examined and compared with three new proposed rule-based algorithms.

3. **To improve the quality of rule sets by developing rule merging techniques for predictive rules and minimising loss of accuracy.**
   The key challenge in predictive rule-based algorithms lies in the need of fewer expressive and accurate rules, which is even more challenging in ensemble systems. After constructing multiple base classifiers induced by a rule-based algorithm, a post-processing process has been applied to improve the quality of the rules by making them more compact without loss of information.

4. **To develop an expressive ensemble learner footed upon the base classifier developed in objective 2 and the rule merging techniques in objective 3.**
   In addition to the high predictive accuracy, which is the main advantage of ensemble learners, a further consideration is required in the ensembles proposed in this thesis, which is the high level of expressiveness. More specifically, using the merging techniques developed in (3) to improve the quality of the rules created by the base learning algorithm developed in (2) makes the ensembles more explainable and computationally efficient.

7.2.2 Contributions to Knowledge

The main contributions of the research presented in this thesis were presented in Chapter 1. These are restated here as follows:

1. A novel classification rule induction approach, called G-Prism, which makes use of Gauss Probability Density Distribution (GPDD) and z-score distribution to generate a new rule-term structure that improves classification performance of the Prism family of algorithms. The new approach produces more expressive and computationally efficient numeric rule-terms compared with converting continuous attributes into categorical ones in the form of frequent discrete intervals. The first version of G-Prism was termed **(G-Prism-FB)** and it has been introduced in [19], a paper published in the 36^{th} SGAI International Conference on Artificial Intelligence.

2. A novel classification rule induction approach, called **G-Prism-DB**, which is a second version of G-Prism algorithm that enables it to expand the coverage of each numeric rule-term, and thus produces fewer rules which are less prone to overfitting. The algorithm is based on a new dynamic rule-term boundaries approach to improve the expressiveness of the rules induced. The dynamic sized boundary can be defined by the user. The work has been introduced in [20], a paper published in the 37^{th} SGAI International Conference on Artificial Intelligence.

3. A novel predictive rule induction algorithm, called **G-Rules-IQR**, which incorporates two new methods in its construction:

   - A new more efficient method in learning heuristics to induce numerical rule-terms directly from continuous attributes based on a combination of: GPDD function, quartiles, and Interquartile Range (IQR). The new method enables producing more compact, accurate and expressive rules using IQR boundaries instead of user defined boundaries.

- A new way to address the challenges in assuming normally distributed attributes in the previous GPDD rule learning algorithms by reducing the skewness rate of numerical attribute values from the normal distribution. G-Rules-IQR algorithm incorporates a prior testing for normality for each attribute in the dataset before applying the approximate normal transformation on the attribute's values.

The work has been introduced in [21], a paper published in the 17^{th} IEEE International Conference on Machine Learning and Applications.

4. A novel framework for ensemble rule-based system, called **Ranked Ensemble G-Rules (ReG-Rules)** learner, which utilises G-Rules-IQR algorithm as a learning algorithm for its base classifiers. This ensemble model provides an approach to harvest the predictive power of an ensemble learner, while maintaining several explainable aspects of rule-based predictive models. ReG-Rules incorporates three novel methods in its construction:

   - A new **Ranking-based method** to rank the base classifiers according to a number of criteria, not only to their accuracies.
   - A new **local Rule Merging (RM) technique** that can reduce the number of rules induced within each individual base classifier without sacrificing the overall predictive accuracy of the model. The approach can be considered as a useful aid in improving the quality of the induced rules and thus developing more expressive rule learners.
   - A new combination technique called **'ReG-Rules Committees'**, which make uses of a weighted voting strategy to decide the final ensemble predictions. The method addresses the potential problem of reliability when some base models are more reliable than others.

   This work has been introduced in [22], a paper published in IEEE Access Journal.

5. A significant extension of the ensemble ReG-Rules learner that can be more expressive while benefiting from the high predictive performance of ensemble learning compared with its stand-alone base classifiers. This system, which is called **'Consolidated Rules Construction (*CRC*)'** incorporates the following novel method in its construction:

   - A new rule consolidation approach, termed **'CRC Consolidator'**, which can compress multiple classifiers' rule sets into one global rule set that can be used directly in predictions without the need to build a committee of rules for each new classification attempt.

6. All the aforementioned algorithms have been implemented and empirically evaluated. All the source codes, which were written in the statistical programming language R are available in online repository at - `https://github.com/ManalAlmutairi/PhD_Project_Codes/tree/v1.0.0` and are archived at - `https://doi.org/10.5281/zenodo.5557590` [23].

7.2.3 Publications

The publications that have been produced during this project are listed below:

- Almutairi, Manal, Frederic Stahl, Mathew Jennings, Thien Le, and Max Bramer. "Towards expressive modular rule induction for numerical attributes." In International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp. 229-235. Springer, Cham, 2016.

- Almutairi, Manal, Frederic Stahl, and Max Bramer. "Improving modular classification rule induction with g-Prism using dynamic rule term boundaries." In International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp. 115-128. Springer, Cham, 2017.

- Almutairi, Manal, Frederic Stahl, and Max Bramer. "A rule-based classifier with accurate and fast rule term induction for continuous attributes." In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 413-420. IEEE, 2018.

- Almutairi, Manal, Frederic Stahl, and Max Bramer. "ReG-Rules: An Explainable Rule-Based Ensemble Learner for Classification." IEEE Access 9 (2021): 52015-52035.

7.2.4 Research Limitations

Given that the focus of this project was on improving the explainability of a predictive ensemble learner, it was not investigated how diversity among individual classifiers contributes to overall ensemble accuracy. The current work presented in this thesis was restricted by the methodology of constructing 'homogeneous ensembles' using a single base learning algorithm; i.e. producing learners of the same type. However, there is potential that the overall accuracy of the ensemble may be further improved by implementing further types of expressive rule-based learners within the same ensemble. Nevertheless, an improved classification accuracy to a standalone rule-based classifiers was achieved. Furthermore, the experimental studies presented in this thesis were not specifically designed to deal with massive

datasets. Though, the ReG-Rules base learners inductions are independent from each other, and thus a parallel version of ReG-Rules is possible to be designed to speed up the training of expressive ensemble models. However, parallel ensemble learning was outside the scope of this research.

7.3 Future Directions

The research presented in this thesis has indicated a number of opportunities to widen the scope of the investigation into various aspects. These are presented here as potential future work to expand upon this thesis further, or as an independent offshoot of this investigation.

7.3.1 Alternative Diversity Generators

The performance of an ensemble model is highly depended on the level of diversity among the group of classifiers that constitute the ensemble. Generating a set of base classifiers should be as diverse as possible to assure producing uncorrelated errors and then obtain a more accurate ensemble [24]. The method that has been adopted in this research and illustrated in Chapter 5 was based on manipulating the training samples using bagging, which is a widely used method in data mining. Then these diverse samples are used to train multiple base classifiers using the same inducer (learning algorithm). This method was robust, efficient and successfully integrated in the ensembles proposed in this thesis.

However, investigating more diversity generators can be considered a further fruitful avenue for future research. For example, (1) diversify individual learners by using different parameter settings for the base learning algorithm. (2) Utilise multi-inducer method, i.e. diversity is obtained by using different types of inducers (multiple induction algorithms). (3) Utilise a hybrid approach based on multi-strategy ensemble learning to combine several ensemble strategies.

7.3.2 Scaling up Ensemble ReG-Rules Learner to Large Data Volumes

With the rapid growth in the amount of data collected by information systems, the need to be able to train a classifier on a massive dataset within a reasonable amount of time is highly demanded. Random sampling might be a solution in some cases, but this often goes at the expense of the model's accuracy. Ensemble learning methods can improve the overall accuracy of the model, but this often goes on the expense of the model's computational efficiency in terms of time and

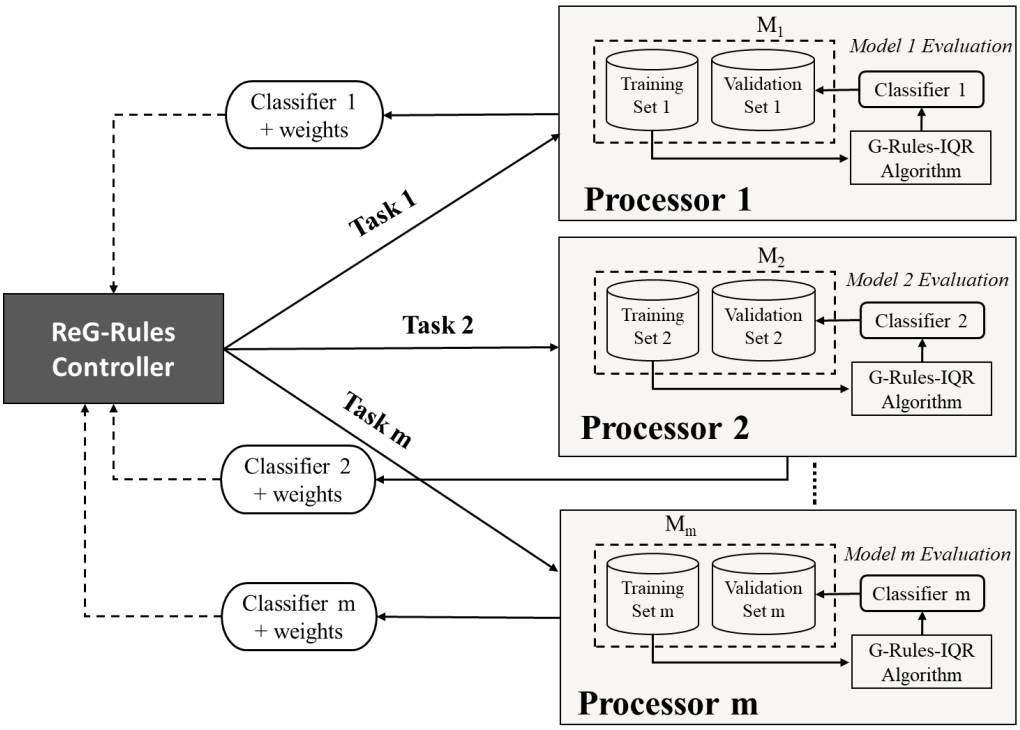


Figure 7.2: Parallelising the Base classifiers Inductions (Stage 2) in ReG-Rules Ensemble Learner

space. Therefore, developing methods to address such issues have become very important. Amongst these methods is building ensemble systems in different computing environments, such as parallel processing and distributed approaches.

Parallel computing can considerably improve the computational efficiency of ensemble systems on high volume data using a form of multiprocessor architectures. Therefore, investigating and developing an appropriate parallel processing method to ReG-Rules learning system can be considered a promising area of future research. This is expected to allow the explainable ensemble ReG-Rules to efficiently scale up to massive datasets. For example, Figure 7.2 shows a suggested framework about how to parallelise the second stage of ReG-Rules, which involves generating independently multiple base classifiers using the same base learning algorithm (see Chapter 5). As can be seen in the figure, each processor can be used to build a single or a number of base classifiers. Hence, many base models can be produced at the same time by executing the same portion of code concurrently on several processors instead of iteratively working on the same machine.

Furthermore, distributed computing can make the ensemble rule based systems (such as ReG-Rules) even more powerful in classifying many new instances simultaneously. This can distribute the computational workload and allow obtaining fast expressive predictions, which considered to be very beneficial in many practical and critical applications.

7.4 Summary of the Chapter

This chapter has summarised the work presented in this thesis in the first section. Then, in Section 7.2, the contributions to knowledge from this project have been described. In the same section, the road map to meet the research aim and objectives has been demonstrated in Figure 7.1 and also the steps taking to achieve them have been briefly summarised. The final section has identified some potential avenues of research for further investigation and development.

Bibliography

[1] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[2] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[3] J. Fürnkranz, D. Gamberger, and N. Lavrač, *Foundations of rule learning*. Springer Science & Business Media, 2012.

[4] G. Dong and J. Bailey, *Contrast Data Mining: Concepts, Algorithms, and Applications*, ser. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, 2016. [Online]. Available: https://books.google.co.uk/books?id=60TNBQAAQBAJ

[5] M. Bramer, *Principles of data mining*. Springer, 2016, vol. 530.

[6] B. Johnston and I. Mathur, *Applied Supervised Learning with Python: Use scikit-learn to build predictive models from real-world datasets and prepare yourself for the future of machine learning*. Packt Publishing, 2019. [Online]. Available: https://books.google.co.uk/books?id=I_eVDwAAQBAJ

[7] A. A. Freitas, "Comprehensible classification models: a position paper," *ACM SIGKDD explorations newsletter*, vol. 15, no. 1, pp. 1–10, 2014.

[8] I. H. Witten and E. Frank, "Data mining: Practical machine learning tools and techniques with java implementations morgan kaufmann," *San Francisco, CA*, 1999.

[9] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[10] J. Koronacki, Z. Ras, and S. Wierzchon, *Advances in Machine Learning I: Dedicated to the Memory of Professor Ryszard S. Michalski*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2010. [Online]. Available: https://books.google.co.uk/books?id=pW3SpsVZiF4C

[11] T. Le, F. Stahl, M. M. Gaber, J. B. Gomes, and G. Di Fatta, "On expressiveness and uncertainty awareness in rule-based classification for data streams," *Neurocomputing*, vol. 265, pp. 127–141, 2017.

[12] M. Saunders, P. Lewis, and A. Thornhill, "Understanding research philosophies and approaches," *Research Methods for Business Students*, vol. 4, pp. 106–135, 01 2009.

[13] D. De Vaus, *Analyzing social science data: 50 key problems in data analysis*. sage, 2002.

[14] E. Fossey, C. Harvey, F. McDermott, and L. Davidson, "Understanding and evaluating qualitative research," *Australian & New Zealand Journal of Psychiatry*, vol. 36, no. 6, pp. 717–732, 2002.

[15] M. M. Ayash, "Research methodologies in computer science and information systems," *Computer Science*, vol. 2014, pp. 1–4, 2014.

[16] A. Newell and H. A. Simon, "Computer science as empirical inquiry: Symbols and search," in *ACM Turing award lectures*, 2007, p. 1975.

[17] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[18] S. Hettich and S. D. Bay, "The uci kdd archive," 1999. [Online]. Available: http://kdd.ics.uci.edu

[19] M. Almutairi, F. Stahl, M. Jennings, T. Le, and M. Bramer, "Towards expressive modular rule induction for numerical attributes," in *Research and Development in Intelligent Systems XXXIII: Incorporating Applications and Innovations in Intelligent Systems XXIV*. Springer, 2016, pp. 229–235.

[20] M. Almutairi, F. Stahl, and M. Bramer, "Improving modular classification rule induction with g-prism using dynamic rule term boundaries," in *Research and Development in Intelligent Systems XXXIII: Incorporating Applications and Innovations in Intelligent Systems XXIV*. Springer, 2017.

[21] ——, "A rule-based classifier with accurate and fast rule term induction for continuous attributes," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 413–420.

[22] ——, "Reg-rules: An explainable rule-based ensemble learner for classification," *IEEE Access*, vol. 9, pp. 52 015–52 035, 2021.

[23] M. K. Almutairi, "ManalAlmutairi/PhD_Project_Codes: G-Rules-IQR, ReG-Rules and CRC," Oct. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5557590

[24] L. Rokach, *Ensemble learning: Pattern classification using ensemble methods*. World Scientific, 2019, vol. 85.

[25] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014.

[26] S. K. Murthy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data mining and knowledge discovery*, vol. 2, no. 4, pp. 345–389, 1998.

[27] J. Quinlan, "Induction of decision trees. mach. learn," 1986.

[28] C. C. Aggarwal, *Data mining: the textbook*. Springer, 2015.

[29] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.

[30] J. R. Quinlan, *C4.5: programs for machine learning*. Elsevier, 2014.

[31] V. N. Vapnik, "The nature of statistical learning," *Theory*, 1995.

[32] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Pearson Education India, 2016.

[33] A. R. Webb, *Statistical pattern recognition*. John Wiley & Sons, 2003.

[34] C. Aggarwal, *Data Classification: Algorithms and Applications*, ser. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, 2015. [Online]. Available: https://books.google.co.uk/books?id=nwQZCwAAQBAJ

[35] L. Rokach, *Pattern classification using ensemble methods*. World Scientific, 2010, vol. 75.

[36] E. Fix, *Discriminatory analysis: nonparametric discrimination, consistency properties*. USAF School of Aviation Medicine, 1951.

[37] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," vol. 2888, 01 2003, pp. 986–996.

[38] D. O. Hebb, *The organisation of behaviour: a neuropsychological theory*. Science Editions New York, 1949.

[39] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[40] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[41] S. Haykin, *Kalman filtering and neural networks*. John Wiley & Sons, 2004, vol. 47.

[42] E. B. Hunt, J. Marin, and P. J. Stone, "Experiments in induction." 1966.

[43] J. Cendrowska, "Prism: An algorithm for inducing modular rules," *International Journal of Man-Machine Studies*, vol. 27, no. 4, pp. 349–370, 1987.

[44] J. R. Quinlan, "Generating production rules from decision trees." in *ijcai*, vol. 87. Citeseer, 1987, pp. 304–307.

[45] ——, "Simplifying decision trees," *International journal of man-machine studies*, vol. 27, no. 3, pp. 221–234, 1987.

[46] R. Kohavi and J. R. Quinlan, "Data mining tasks and methods: Classification: decision-tree discovery," in *Handbook of data mining and knowledge discovery*, 2002, pp. 267–276.

[47] L. Breiman, J. Friedman, R. Olshen, and C. J. Stone, "Classification and regression trees," 1983.

[48] T. Oates and D. Jensen, "The effects of training set size on decision tree complexity," in *Proc. 14th Int. Conf. on Machine Learning*. Citeseer, 1997.

[49] K. Grąbczewski, "Techniques of decision tree induction," in *Meta-Learning in Decision Tree Induction*. Springer, 2014, pp. 11–117.

[50] R. S. Michalski, "On the quasi-minimal solution of the general covering problem," 1969.

[51] G. Bagallo and D. Haussler, "Boolean feature discovery in empirical learning," *Machine learning*, vol. 5, no. 1, pp. 71–99, 1990.

[52] T. M. Mitchell *et al.*, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, no. 37, pp. 870–877, 1997.

[53] R. S. Michalski, "Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of development an expert system for soybean disease diagnosis," *International Journal of Policy Analysis and Information Systems*, vol. 4, no. 2, pp. 125–161, 1980.

[54] 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," *Proc. AAAI 1986*, pp. 1–041, 1986.

[55] P. Clark and T. Niblett, "The cn2 induction algorithm," *Machine learning*, vol. 3, no. 4, pp. 261–283, 1989.

[56] W. W. Cohen, "Fast effective rule induction," in *Machine learning proceedings 1995*. Elsevier, 1995, pp. 115–123.

[57] F. Bergadano, S. Matwin, R. S. Michalski, and J. Zhang, "Learning two-tiered descriptions of flexible concepts: The poseidon system," *Machine Learning*, vol. 8, no. 1, pp. 5–43, 1992.

[58] K. A. Kaufman and R. S. Michalski, "An adjustable description quality measure for pattern discovery using the aq methodology," *Journal of Intelligent Information Systems*, vol. 14, no. 2-3, pp. 199–216, 2000.

[59] R. S. Michalski, "Pattern recognition as rule-guided inductive inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 4, pp. 349–361, 1980.

[60] R. Michalski and J. Larson, "Selection of most representative training examples and incremental addition of vl1 hypothesis: The underlying methodology and the description of programs esel and aq11," Technical Report, Tech. Rep., 1978.

[61] G. Cervone, P. Franzese, and A. P. Keesee, "Algorithm quasi-optimal (aq) learning," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 2, pp. 218–236, 2010.

[62] P. Clark and R. Boswell, "Rule induction with cn2: Some recent improvements," in *European Working Session on Learning*. Springer, 1991, pp. 151–163.

[63] S. Dzeroski and I. Bratko, "Handling noise in inductive logic programming," in *Proceedings of the International Workshop on Inductive Logic Programming*, vol. 91, 1992.

[64] D. Raedt, *Advances in inductive logic programming*. IOS press, 1996.

[65] H. Theron and I. Cloete, "Bexa: A covering algorithm for learning propositional concept descriptions," *Machine Learning*, vol. 24, no. 1, pp. 5–40, 1996.

[66] J. Fürnkranz and G. Widmer, "Incremental reduced error pruning," in *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 70–77.

[67] J. Hühn and E. Hüllermeier, "Furia: an algorithm for unordered fuzzy rule induction," *Data Mining and Knowledge Discovery*, vol. 19, no. 3, pp. 293–319, 2009.

[68] A. Palacios, L. Sánchez, I. Couso, and S. Destercke, "An extension of the furia classification algorithm to low quality data through fuzzy rankings and its application to the early diagnosis of dyslexia," *Neurocomputing*, vol. 176, pp. 60–71, 2016.

[69] M. Bramer, "Automatic induction of classification rules from examples using n-prism," *Research and development in intelligent systems XVI*, pp. 99–121, 2000.

[70] ——, "Inducer: a rule induction workbench for data mining," in *Proceedings of the 16th IFIP World Computer Congress Conference on Intelligent Information Processing, Publishing House of Electronics Industry, Beijing*. Citeseer, 2000, pp. 499–506.

[71] ——, "Inducer: a public domain workbench for data mining," *International Journal of Systems Science*, vol. 36, no. 14, pp. 909–919, 2005.

[72] F. Stahl and M. Bramer, "Computationally efficient induction of classification rules with the pmcri and j-pmcri frameworks," *Knowledge-Based Systems*, vol. 35, pp. 49–63, 2012.

[73] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1, pp. 1–39, 2010.

[74] W. Leigh, R. Purvis, and J. M. Ragusa, "Forecasting the nyse composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: a case study in romantic decision support," *Decision support systems*, vol. 32, no. 4, pp. 361–377, 2002.

[75] D. Liang, C.-F. Tsai, A.-J. Dai, and W. Eberle, "A novel classifier ensemble approach for financial distress prediction," *Knowledge and Information Systems*, vol. 54, no. 2, pp. 437–462, 2018.

[76] S.-K. S. Fan, C.-Y. Hsu, D.-M. Tsai, F. He, and C.-C. Cheng, "Data-driven approach for fault detection and diagnostic in semiconductor manufacturing," *IEEE Transactions on Automation Science and Engineering*, 2020.

[77] Z. Wang, R. Wang, J. Gao, Z. Gao, and Y. Liang, "Fault recognition using an ensemble classifier based on dempster–shafer theory," *Pattern Recognition*, vol. 99, p. 107079, 2020.

[78] F. Ghanbari-Adivi and M. Mosleh, "Text emotion detection in social networks using a novel ensemble classifier based on parzen tree estimator (tpe)," *Neural Computing and Applications*, vol. 31, no. 12, pp. 8971–8983, 2019.

[79] J. D. Gupta, S. Samanta, and B. Chanda, "Ensemble classifier-based offline handwritten word recognition system in holistic approach," *IET Image Processing*, vol. 12, no. 8, pp. 1467–1474, 2018.

[80] P. Pławiak and U. R. Acharya, "Novel deep genetic ensemble of classifiers for arrhythmia detection using ecg signals," *Neural Computing and Applications*, vol. 32, no. 15, pp. 11 137–11 161, 2020.

[81] M. Muzammal, R. Talat, A. H. Sodhro, and S. Pirbhulal, "A multi-sensor data fusion enabled ensemble approach for medical data from body sensor networks," *Information Fusion*, vol. 53, pp. 155–164, 2020.

[82] M. Anbarasi and M. S. Durai, "Incipient knowledge in protein folding kinetics states prophecy using deep neural network-based ensemble classifier," *International Journal of Computer Aided Engineering and Technology*, vol. 13, no. 3, pp. 341–359, 2020.

[83] G. Idakwo, S. Thangapandian, J. Luttrell, Y. Li, N. Wang, Z. Zhou, H. Hong, B. Yang, C. Zhang, and P. Gong, "Structure–activity relationship-based chemical classification of highly imbalanced tox21 datasets," *Journal of Cheminformatics*, vol. 12, no. 1, pp. 1–19, 2020.

[84] A. Arabameri, S. Saha, J. Roy, J. P. Tiefenbacher, A. Cerda, T. Biggs, B. Pradhan, P. T. T. Ngo, and A. L. Collins, "A novel ensemble computational intelligence approach for the spatial prediction of land subsidence susceptibility," *Science of The Total Environment*, p. 138595, 2020.

[85] A. Datta and R. Chatterjee, "Comparative study of different ensemble compositions in eeg signal classification problem," in *Emerging Technologies in Data Mining and Information Security*. Springer, 2019, pp. 145–154.

[86] B. Fazzinga, F. Folino, F. Furfaro, and L. Pontieri, "An ensemble-based approach to the security-oriented classification of low-level log traces," *Expert Systems with Applications*, p. 113386, 2020.

[87] C. Ju, A. Bibaut, and M. van der Laan, "The relative performance of ensemble methods with deep convolutional neural networks for image classification," *Journal of Applied Statistics*, vol. 45, no. 15, pp. 2800–2818, 2018.

[88] V. P. Singh and R. Srivastava, "Improved image retrieval using fast colour-texture features with varying weighted similarity measure and random forests," *Multimedia Tools and Applications*, vol. 77, no. 11, pp. 14 435–14 460, 2018.

[89] R. Logesh, V. Subramaniyaswamy, D. Malathi, N. Sivaramakrishnan, and V. Vijayakumar, "Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method," *Neural Computing and Applications*, vol. 32, no. 7, pp. 2141–2164, 2020.

[90] H. Wang, "Utilizing imbalanced data and classification cost matrix to predict movie preferences," *arXiv preprint arXiv:1812.02529*, 2018.

[91] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*. Springer, 2000, pp. 1–15.

[92] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[93] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[94] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[95] T. Bylander and L. Tate, "Using validation sets to avoid overfitting in adaboost." in *Flairs conference*, 2006, pp. 544–549.

[96] K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan, *Data mining: a knowledge discovery approach*. Springer Science & Business Media, 2007.

[97] J. Wirth and J. Catlett, "Experiments on the costs and benefits of windowing in id3," in *Machine Learning Proceedings 1988*. Elsevier, 1988, pp. 87–99.

[98] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[99] A. Dogan and D. Birant, "A weighted majority voting ensemble approach for classification," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*. IEEE, 2019, pp. 1–6.

[100] K. Tumer and J. Ghosh, "Error correlation and error reduction in ensemble classifiers," *Connection science*, vol. 8, no. 3-4, pp. 385–404, 1996.

[101] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[102] T. K. Ho, "Random decision forests," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1. IEEE, 1995, pp. 278–282.

[103] ——, "The random subspace method for constructing decision forests," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[104] J. Gareth, W. Daniela, H. Trevor, and T. Robert, *An introduction to statistical learning: with applications in R*. Spinger, 2013.

[105] P. Probst, M. N. Wright, and A.-L. Boulesteix, "Hyperparameters and tuning strategies for random forest," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 3, p. e1301, 2019.

[106] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–36, 2018.

[107] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[108] F. Stahl and M. Bramer, "Random prism: a noise-tolerant alternative to random forests," *Expert Systems*, vol. 31, no. 5, pp. 411–420, 2014.

[109] M. Bramer, "An information-theoretic approach to the pre-pruning of classification rules," in *International Conference on Intelligent Information Processing*. Springer, 2002, pp. 201–212.

[110] F. Stahl, D. May, H. Mills, M. Bramer, and M. M. Gaber, "A scalable expressive ensemble learning using random prism: A mapreduce approach," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XX*. Springer, 2015, pp. 90–107.

[111] F. Stahl, D. May, and M. Bramer, "Parallel random prism: a computationally efficient ensemble learner for classification," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 2012, pp. 21–34.

[112] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[113] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2011.

[114] H. Bonab and F. Can, "Less is more: a comprehensive framework for the number of components of ensemble classifiers," *IEEE Transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2735–2745, 2019.

[115] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: a survey and categorisation," *Information fusion*, vol. 6, no. 1, pp. 5–20, 2005.

[116] W. L. Buntine, "A theory of learning classification rules," Ph.D. dissertation, Citeseer, 1990.

[117] A. P. Dempster *et al.*, "Upper and lower probabilities generated by a random closed interval," *The Annals of Mathematical Statistics*, vol. 39, no. 3, pp. 957–966, 1968.

[118] G. Shafer, *A mathematical theory of evidence*. Princeton university press, 1976, vol. 42.

[119] S. Shlien, "Multiple binary decision tree classifiers," *Pattern Recognition*, vol. 23, no. 7, pp. 757–763, 1990.

[120] S. Razi, M. R. K. Mollaei, and J. Ghasemi, "A novel method for classification of bci multi-class motor imagery task based on dempster–shafer theory," *Information Sciences*, vol. 484, pp. 14–26, 2019.

[121] D. Hernández-Lobato, G. Martínez-Muñoz, and A. Suárez, "How large should ensembles of classifiers be?" *Pattern Recognition*, vol. 46, no. 5, pp. 1323–1336, 2013.

[122] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?" in *International workshop on machine learning and data mining in pattern recognition*. Springer, 2012, pp. 154–168.

[123] H. Liu, A. Mandvikar, and J. Mody, "An empirical study of building compact ensembles," in *International Conference on Web-Age Information Management*. Springer, 2004, pp. 622–627.

[124] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[125] G. Tsoumakas, I. Partalas, and I. Vlahavas, "A taxonomy and short review of ensemble selection," in *Workshop on Supervised and Unsupervised Ensemble Methods and Their Applications*, 2008, pp. 1–6.

[126] D. Tripathi, D. R. Edla, R. Cheruku, and V. Kuppili, "A novel hybrid credit scoring model based on ensemble feature selection and multilayer ensemble classification," *Computational Intelligence*, vol. 35, no. 2, pp. 371–394, 2019.

[127] A. L. Prodromidis, S. J. Stolfo, and P. K. Chan, "Effective and efficient pruning of meta-classifiers in a distributed data mining system," *Knowledge Discovery and Data Mining Journal. submitted for publication*, vol. 32, 1999.

[128] Q. Hu, D. Yu, Z. Xie, and X. Li, "Eros: Ensemble rough subspaces," *Pattern recognition*, vol. 40, no. 12, pp. 3728–3739, 2007.

[129] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *ICML*, vol. 97. Citeseer, 1997, pp. 211–218.

[130] G. Martínez-Muñoz and A. Suárez, "Pruning in ordered bagging ensembles," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 609–616.

[131] R. Arbel and L. Rokach, "Classifier evaluation under limited resources," *Pattern Recognition Letters*, vol. 27, no. 14, pp. 1619–1631, 2006.

[132] A. Cano, A. Zafra, and S. Ventura, "An interpretable classification rule mining algorithm," *Information Sciences*, vol. 240, pp. 1–20, 2013.

[133] M. Bramer, *Principles of data mining*. Springer, 2007, vol. 180.

[134] U. Fayyad and K. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," 1993.

[135] K. Lavangnananda and S. Chattanachot, "Study of discretization methods in classification," in *2017 9th International Conference on Knowledge and Smart Technology (KST)*. IEEE, 2017, pp. 50–55.

[136] S. Kotsiantis and D. Kanellopoulos, "Discretization techniques: A recent survey," *GESTS International Transactions on Computer Science and Engineering*, vol. 32, no. 1, pp. 47–58, 2006.

[137] B. Hemada and K. Lakshmi, "A study on discretization techniques," *Int. J. of Engineering Research & Technology*, vol. 2, no. 8, pp. 1887–1892, 2013.

[138] R. Kerber, "Chimerge: Discretization of numeric attributes," in *Proceedings of the tenth national conference on Artificial intelligence*. Aaai Press, 1992, pp. 123–128.

[139] L. A. Kurgan and K. J. Cios, "Caim discretization algorithm," *IEEE transactions on Knowledge and Data Engineering*, vol. 16, no. 2, pp. 145–153, 2004.

[140] J. Quinlan, "5 - probabilistic decision trees," in *Machine Learning*, Y. Kodratoff and R. S. Michalski, Eds. San Francisco (CA): Morgan Kaufmann, 1990, pp. 140–152. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080510552500110

[141] C. Lee, C. Tsai, Y. Yang, and W. Yang, "A top-down and greedy method for discretization of continuous attributes," in *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, vol. 1, 2007, pp. 472–476.

[142] C.-J. Tsai, C.-I. Lee, and W.-P. Yang, "A discretization algorithm based on class-attribute contingency coefficient," *Information Sciences*, vol. 178, no. 3, pp. 714–731, 2008, including Special Issue "Ambient Intelligence". [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025507004148

[143] C. Walck *et al.*, "Hand-book on statistical distributions for experimentalists," *University of Stockholm*, vol. 10, 2007.

[144] D. Lane, “Introduction to Normal Distributions,” 1 2021, [Online; accessed 2021-04-03]. [Online]. Available: https://chem.libretexts.org/@go/page/2117

[145] K. M. Ramachandran and C. P. Tsokos, *Mathematical statistics with applications in R*. Academic Press, 2020.

[146] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[147] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2020, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org

[148] H. C. Thode, *Testing for normality*. CRC press, 2002, vol. 164.

[149] D. G. Altman and J. M. Bland, “Statistics notes: the normal distribution,” *Bmj*, vol. 310, no. 6975, p. 298, 1995.

[150] P. Mishra, C. M. Pandey, U. Singh, A. Gupta, C. Sahu, and A. Keshri, “Descriptive statistics and normality tests for statistical data,” *Annals of cardiac anaesthesia*, vol. 22, no. 1, p. 67, 2019.

[151] S. S. Shapiro and M. B. Wilk, “An analysis of variance test for normality (complete samples),” *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[152] C. M. Jarque and A. K. Bera, “Efficient tests for normality, homoscedasticity and serial independence of regression residuals,” *Economics letters*, vol. 6, no. 3, pp. 255–259, 1980.

[153] A. Ghasemi and S. Zahediasl, “Normality tests for statistical analysis: a guide for non-statisticians,” *International journal of endocrinology and metabolism*, vol. 10, no. 2, p. 486, 2012.

[154] B. Yazici and S. Yolacan, “A comparison of various tests of normality,” *Journal of Statistical Computation and Simulation*, vol. 77, no. 2, pp. 175–183, 2007.

[155] S. Amari *et al.*, *The handbook of brain theory and neural networks*. MIT press, 2003.