New Rule Induction Algorithms with Improved Noise Tolerance and Scalability

A thesis submitted to Cardiff University

for the degree of

Doctor of Philosophy

by

Khurram Shehzad

Intelligent Systems Research Laboratory

Manufacturing Engineering Centre

Cardiff University

United Kingdom

2009

ABSTRACT

As data storage capacities continue to increase due to rapid advances in information technology, there is a growing need for devising scalable data mining algorithms able to sift through large volumes of data in a short amount of time. Moreover, real-world data is inherently imperfect due to the presence of noise as opposed to artificially prepared data. Consequently, there is also a need for designing robust algorithms capable of handling noise, so that the discovered patterns are reliable with good predictive performance on future data. This has led to ongoing research in the field of data mining, intended to develop algorithms that are scalable as well as robust.

The most straightforward approach for handling the issue of scalability is to develop efficient algorithms that can process large datasets in a relatively short time. Efficiency may be achieved by employing suitable rule mining constraints that can drastically cut down the search space. The first part of this thesis focuses on the improvement of a state-of-the-art rule induction algorithm, RULES-6, which incorporates certain search space pruning constraints in order to scale to large datasets. However, the constraints are insufficient and also have not been exploited to the maximum, resulting in the generation of specific rules which not only increases learning time but also the length of the rule set. In order to address these issues, a new algorithm RULES-7 is proposed which uses deep rule mining constraints from association learning. This results in a significant drop in execution time for large datasets while boosting the classification accuracy of the model on future data. A novel comparison heuristic is also proposed for the algorithm which improves classification accuracy while maintaining the execution time.

Since an overwhelming majority of induction algorithms are unable to handle the continuous data ubiquitous in the real-world, it is also necessary to develop an efficient discretisation procedure whereby continuous attributes can be treated as discrete. By generalizing the raw continuous data, discretisation helps to speed up the induction process and results in a simpler and intelligible model that is also more accurate on future data. Many preprocessing discretisation techniques have been proposed to date, of which the entropy based technique has by far been accepted as the most accurate. However, the technique is suboptimal for classification because of failing to identify the cut points within the value range of each class for a continuous attribute, which deteriorates its classification accuracy. The second part of this thesis presents a new discretisation technique which utilizes the entropy based principle but takes a class-centered approach to discretisation. The proposed technique not only increases the efficiency of rule induction but also improves the classification accuracy of the induced model.

Another issue with existing induction algorithms relates to the way covered examples are dealt with when a new rule is formed. To avoid problems such as fragmentation and small disjuncts, the RULES family of algorithms marks the examples instead of removing them. This tends to increase overlapping between rules. The third part of this thesis proposes a new hybrid pruning technique in order to address the overlapping issue so as to reduce the rule set size. It also proposes an incremental post-pruning technique designed specifically to handle the issue of noisy data. This leads to improved induction performance as well as better classification accuracy.

In The Name of Allah,

The Most Gracious, The Most Merciful

ACKNOWLEDGEMENTS

At the outset of my list of acknowledgements, I must offer my sincere gratitude to the Almighty, who not only gave me the opportunity to undertake research in the UK but also helped me immensely right from the start up until completion of my work.

I consider myself fortunate to have Professor D. T. Pham as my supervisor, and I'm indebted for his tireless support during the past three years of my study at Cardiff University. He has always been very keen to see me from time to time in order to discuss my progress and to give useful suggestions for subsequent work. On top of this, he is also a very nice and jolly person who loves to work with his students in a friendly atmosphere. His consistent encouragement has always boosted my morale and urged me to work even harder and produce better results.

Thanks are also due to Cardiff University for their great student support in all spheres of academic life. In particular I would like to offer my thanks to the Intelligent Systems Laboratory, Systems Engineering Division, Cardiff School of Engineering for providing a conducive study environment and computing facilities to pursue this research.

I must also offer my grateful thanks to my sponsoring university UET back in Pakistan for their funding under the *Faculty Development Program* (FDP) scholarship, without which it would have been impossible for me to study in the UK. I would also like to offer words of appreciation to all my colleagues and seniors at the Manufacturing Engineering Centre for their friendly attitude and continual support.

CONTENTS

ABSTRACT ii

ACKNOWLEDGEMENTS v

DECLARATION vi

CONTENTS vii

LIST OF FIGURES xii

LIST OF TABLES xv

ABBREVIATIONS xvii

SYMBOLS xviii

CHAPTER 1 INTRODUCTION **1**

1.1 Motivation 1

1.2 Research Objectives 5

1.3 Research Methodology 6

1.4 Thesis Outline 7

CHAPTER 2 SURVEY OF CLASSIFICATION LEARNING AND RELATED FIELDS **9**

2.1 Foreword 9

2.2 Fundamentals of Classification Learning 11

2.2.1 Format of Training Data 14

2.2.2 Model Representation 16

2.2.2.1 Decision Tree Induction 16

2.2.2.2 Rule Induction 21

2.2.3 Model Evaluation 23

2.3 Major Inductive Learning Algorithms for Classification 25

2.3.1 Decision Tree Algorithms 25

2.3.2 Rule Induction Algorithms 30

2.3.3 Major Contributions on Scalability Issue 39

2.4 Essentials of Association Learning 45

2.4.1 Frequent Itemset Mining 45

2.4.1.1 Early Frequent Itemset Mining 46

2.4.1.2 Closed and Maximal Frequent Itemset Mining 47

2.4.1.3 Frequent Itemset Mining Without Candidates 50

2.4.2 Generation of Association Rules 51

2.5 New Approaches to Classification Learning 52

2.6 Summary 54

CHAPTER 3 RULES-7: AN EFFICIENT NOISE TOLERANT RULE INDUCTION ALGORITHM 56

3.1 Motivation 56

3.2 The RULES-6 Algorithm 58

3.3 Problems Identified with RULES-6 64

3.3.1 Duplicate Candidate Rules 66

3.3.2 Enhancement of the Control Structure 67

3.3.2.1 Rectification 70

3.3.2.2 Other Efficiency Improvements 72

3.3.3 Specialisation Measure Parameter 73

3.4 The RULES-7 Algorithm 74

3.4.1 The Comparison Heuristic (λ) 74

3.4.2 ParentRule Pruning Technique (MS) 79

3.4.3 ChildRule Pruning Technique (MPS) 82

3.5 Empirical Evaluation of RULES-7 84

3.5.1 RULES-7 vs. RULES-6 86

3.5.2 RULES-7@λ vs. RULES-6 94

3.5.3 RULES-7@MS vs. RULES-6 98

3.5.4 RULES-7@MPS vs. RULES-6 102

3.5.5 Effect of Sample Size 106

3.5.6 Consolidated Results 106

3.6 Summary 113

CHAPTER 4 EDISC: A NEW DISCRETISATION METHOD FOR RULE INDUCTION 116

4.1 Motivation 116

4.2 Taxonomy of Major Discretisation Approaches 119

4.2.1 Unsupervised vs. Supervised 119

4.2.2 Static vs. Dynamic 120

4.2.3 Global vs. Local 120

4.2.4 Parametric vs. Non-parametric 121

4.2.5 Hierarchical vs. Non-hierarchical 121

4.2.6 Disjoint vs. Non-disjoint 122

4.2.7 Fuzzy vs. Non-fuzzy 122

4.3 Description of Techniques Used for Comparison 122

4.3.1 Equal-width Discretisation 122

4.3.2 1R Discretisation 123

4.3.3 Entropy-MDLP Discretisation 124

4.4 Proposed Discretisation Technique 125

4.5 Empirical Evaluation of EDISC 129

4.6 Summary 141

CHAPTER 5 NEW SIMPLE PRUNING TECHNIQUES FOR RULE INDUCTION 145

5.1 Motivation 145

5.2 Survey of Existing Pruning Techniques 147

5.2.1 Pre-pruning Techniques 147

5.2.2 Post-pruning Techniques 148

5.2.3 Hybrid-pruning Techniques 150

5.3 Proposed Pruning Techniques 152

5.3.1 Minimum New Classification (MNC) 152

5.3.2 Permissible Misclassification (PMC) 161

5.4 Empirical Evaluation of Proposed Pruning Techniques 164

5.4.1 RULES-7@MNC vs. RULES-7 167

5.4.2 RULES-7@PMC vs. RULES-7 174

5.4.3 Consolidated Results 174

5.5 Summary 174

CHAPTER 6 CONCLUSION 183

6.1 Contributions 183

6.2 Conclusions 185

6.3 Future Research Directions 187

APPENDIX A: DESCRIPTION OF DATASETS 189

APPENDIX B: THE RULES-3 PLUS ALGORITHM 199

APPENDIX C: THE SRI ALGORITHM 201

REFERENCES 204

.

LIST OF FIGURES

Figure 2.1 The Data Classification Process 12

Figure 2.2 Decision Tree for the Contact Lens Data 18

Figure 2.3 Flowchart Describing the Evolution of RULES Family 40

Figure 2.4 The Concept of Closed and Maximal Frequent Itemsets 48

Figure 2.5 The Itemset Lattice Illustrating Closed and Maximal Frequent Itemsets 49

Figure 3.1 A pseudo-code description of RULES-6 60

Figure 3.2 A pseudo-code description of the *Induce_One_Rule* procedure of RULES-6 61

Figure 3.3 RULES-6 Search Space Pruning Control Structure 68

Figure 3.4 Modified Search Space Pruning Control Structure for RULES-7.............. 71

Figure 3.5 A pseudo-code description of RULES-7 75

Figure 3.6 A pseudo-code description of the *Induce_One_Rule* procedure of RULES-7 76

Figure 3.7 RULES-7 vs. RULES-6 in terms of Number of Rules 92

Figure 3.8 RULES-7 vs. RULES-6 in terms of Accuracy 93

Figure 3.9 RULES-7 vs. RULES-6 in terms of Exec. Time 93

Figure 3.10 RULES-7@λ vs. RULES-6 in terms of Number of Rules 96

Figure 3.11 RULES-7@λ vs. RULES-6 in terms of Accuracy 97

Figure 3.12 RULES-7@λ vs. RULES-6 in terms of Exec. Time 97

Figure 3.13 RULES-7@MS vs. RULES-6 in terms of Number of Rules 100

Figure 3.14 RULES-7@MS vs. RULES-6 in terms of Accuracy 101

Figure 3.15 RULES-7@MS vs. RULES-6 in terms of Exec. Time 101

Figure 3.16 RULES-7@MPS vs. RULES-6 in terms of Number of Rules 104

Figure 3.17 RULES-7@MPS vs. RULES-6 in terms of Accuracy 105

Figure 3.18 RULES-7@MPS vs. RULES-6 in terms of Exec. Time 105

Figure 3.19 Number of Rules vs. Sample Size for Cover-Type Data 108

Figure 3.20 Number of Rules vs. Sample Size for Nursery Data 108

Figure 3.21 Accuracy vs. Sample Size for Cover-Type Data 109

Figure 3.22 Accuracy vs. Sample Size for Nursery Data 109

Figure 3.23 Exec. Time vs. Sample Size for Cover-Type Data 110

Figure 3.24 Exec. Time vs. Sample Size for Nursery Data 110

Figure 3.25 RULES-7 New Techniques Comparison in terms of Number of Rules 113

Figure 3.26 RULES-7 New Techniques Comparison in terms of Accuracy 114

Figure 3.27 RULES-7 New Techniques Comparison in terms of Exec. Time 114

Figure 4.1 Illustration of the concept of *Scope* of Classes 127

Figure 4.2 The EDISC Algorithm 128

Figure 4.3 Modified pseudo-code description of RULES-7 132

Figure 4.4 Modified pseudo-code description of the *Induce_One_Rule* procedure of RULES-7 133

Figure 4.5 The *Find_Interval* procedure 135

Figure 4.6 Discretisation Techniques Comparison in terms of Number of Rules 142

Figure 4.7 Discretisation Techniques Comparison in terms of Accuracy 142

Figure 4.8 Discretisation Techniques Comparison in terms of Disc. Time 143

Figure 4.9 Discretisation Techniques Comparison in terms of Exec. Time 143

Figure 5.1 A pseudo-code description of RULES-7 with MNC 155

Figure 5.2 A pseudo-code description of RULES-7 with MNC^1 and MNC^2............ 158

Figure 5.3 A pseudo-code description of the *Induce_One_Rule* procedure of RULES-7 with *MNC^1* and *MNC^2* 159

Figure 5.4 Modified pseudo-code description of RULES-7 with PMC 163

Figure 5.5 A pseudo-code description of the *Generalise_Rule* procedure 164

Figure 5.6 RULES-7@MNC^1 vs. RULES-7 in terms of Number of Rules 169

Figure 5.7 RULES-7@MNC^1 vs. RULES-7 in terms of Accuracy 170

Figure 5.8 RULES-7@ MNC^1 vs. RULES-7 in terms of Exec. Time 170

Figure 5.9 RULES-7@MNC^2 vs. RULES-7 in terms of Number of Rules 172

Figure 5.10 RULES-7@MNC^2 vs. RULES-7 in terms of Accuracy 173

Figure 5.11 RULES-7@ MNC^2 vs. RULES-7 in terms of Exec. Time 173

Figure 5.12 RULES-7@PMC vs. RULES-7 in terms of Number of Rules 176

Figure 5.13 RULES-7@PMC vs. RULES-7 in terms of Accuracy 177

Figure 5.14 RULES-7@PMC vs. RULES-7 in terms of Exec. Time 177

Figure 5.15 Comparison of Pruning Techniques in terms of Number of Rules 180

Figure 5.16 Comparison of Pruning Techniques in terms of Accuracy 181

Figure 5.17 Comparison of Pruning Techniques in terms of Exec. Time 181

LIST OF TABLES

Table 2.1 The Contact Lens Data (nominal-type) 15

Table 2.2 The Iris Data (continuous-type) 17

Table 2.3 The Weather Data (mixed-type) 17

Table 2.4 An Example Transactional Database with Five Items 48

Table 3.1 The Modified Terminology 63

Table 3.2 The Weather Dataset 65

Table 3.3 Summary of Datasets used in Experiments 85

Table 3.4 RULES-7 vs. RULES-6 with respect to Beam Width 87

Table 3.5 RULES-7 vs. RULES-6 with respect to MinPositives 88

Table 3.6 RULES-7 vs. RULES-6 with respect to MinExcludedNeg 89

Table 3.7 RULES-7 vs. RULES-6 91

Table 3.8 Summary of RULES-7 92

Table 3.9 RULES-7@λ vs. RULES-6 95

Table 3.10 Summary of RULES-7@λ 96

Table 3.11 RULES-7@MS vs. RULES-6 99

Table 3.12 Summary of RULES-7@MS 100

Table 3.13 RULES-7@MPS vs. RULES-6 103

Table 3.14 Summary of RULES-7@MPS 104

Table 3.15 RULES-7@MS vs. RULES-6 with respect to Sample Size 107

Table 3.16 RULES-7@MPS vs. RULES-6 with respect to Sample Size 107

Table 3.17 Consolidated Results 111

Table 3.18 Summary of All Techniques 113

Table 4.1 EDISC Comparison with Entropy_MDLP for *Iris* Data 130

Table 4.2 EDISC Comparison with Entropy_MDLP for *Cover-Type* Data 130

Table 4.3 EDISC Comparison with Entropy_MDLP for *Landsat* Data 131

Table 4.4 EDISC Comparison with Entropy_MDLP for *Letter* Data 131

Table 4.5 Summary of Datasets used in Experiments 137

Table 4.6 Comparison of EDISC against Three Other Discretisation Techniques 138

Table 4.7 Summary of All Discretisation Techniques 140

Table 5.1 Summary of Datasets used in Experiments 166

Table 5.2 RULES-7@MNC^1 vs. RULES-7 168

Table 5.3 Summary of RULES-7@MNC^1 169

Table 5.4 RULES-7@MNC^2 vs. RULES-7 171

Table 5.5 Summary of RULES-7@MNC^2 172

Table 5.6 RULES-7@PMC vs. RULES-7 175

Table 5.7 Summary of RULES-7@PMC 176

Table 5.8 Consolidated Results for New Pruning Techniques 178

Table 5.9 Summary of New Pruning Techniques 180

ABBREVIATIONS

AQ	Algorithm Quasi-optimal
ARM	Association Rule Mining
CART	Classification and Regression Trees
DIC	Dynamic Itemset Counting
DM	Data Mining
EP	Estimate of Probability
FIM	Frequent Itemset Mining
FOIL	First Order Inductive Learner
FP	Frequent Pattern
ID3	Iterative Dichotomiser Version 3
ILP	Inductive Learning Process
IREP	Incremental Reduced Error Pruning
KDD	Knowledge Discovery from Databases
MDL	Minimum Description Length
MF	Measure of Fit
ML	Machine Learning
REP	Reduced Error Pruning
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
RULES-6	RULe Extraction System Version 6
RULES-7	RULe Extraction System Version 7
SRI	Scalable Rule Induction
STM	Short Term Memory
TDP	Top-Down Pruning

SYMBOLS

C_i	-	the *i*th class in the dataset
D	-	a dataset used for rule mining
D_j	-	the *j*th partition of the dataset *D*
e_i	-	expected frequency of examples among classes
E_c	-	Entropy of a cut-point
f_i	-	observed frequency of examples among classes
k	-	the number of classes in a dataset
m	-	a domain dependent noise control parameter
N	-	the total number of examples in a dataset
P_0	-	the probability of target class in the training data
$\sum_c$	-	sum for continuous attributes
$\sum_d$	-	sum for discrete attributes
V_R^i	-	value of the *i*th condition in rule *R*
V_E^i	-	value of the *i*th attribute in example *E*
w	-	a parameter to specify the beam width
ξ	-	the support differentia factor

CHAPTER 1

INTRODUCTION

1.1 Motivation

The extraction of valuable knowledge from raw data is not a new concept and analysts have performed this task manually from limited quantities of stored data in the past. However, due to recent advances in information technology in the last decade, resulting in an ever increasing capacity for storing gigantic quantities of real-world data, the manual knowledge discovery approach is no longer practical. This has led to the development of a burgeoning new technology in the field of computer science known as *data mining (DM)*. Data mining aims to discover valuable knowledge from large volumes of stored data automatically by means of computer algorithms and is consequently also referred to as *knowledge discovery from databases (KDD)*.

Data mining is in fact a large-scale application of *machine learning (ML)*, a research area in artificial intelligence that attempts to mimic human intelligence by designing and developing algorithms and techniques that allow computers to learn. Intelligent behavior in human beings comes through interaction with their environment, the result of which is summarised in the form of a model by means of an *inductive learning process (ILP)*. Machine learning is capable of automating this process of inductive learning by using coded observations from a stored dataset instead of interacting directly with the environment (Holsheimer and Siebes, 1991). The model so obtained identifies the regularities and patterns underlying the dataset in question and can be used to gain insight into the underlying structure for better decision-making as well as to make predictions on future unseen data.

Although machine learning provides the technical basis for data mining, it's the important practical considerations that justify its existence as an independent field. For example, data stored in an extremely large real-world database is always generated and stored for purposes other than learning. Consequently, data cleansing is a major preliminary step in data mining, whereas it has no significance in case of machine learning which takes as an input laboratory prepared data carefully selected for learning purposes. Furthermore, the size of the database makes the verification of the hypotheses an extremely costly process in the case of data mining, so advanced database techniques such as browsing optimisation and caching have to be used in order to reduce these costs. As a result, the knowledge discovery process in data mining is much harder due to the absence of the ideal conditions found in machine learning (Holsheimer and Siebes, 1991). It is also the main reason why a vast majority of the so-called "data mining systems" on the market are unable to perform true data mining and should more appropriately be categorised as either machine learning systems or statistical data analysis tools (Han and Kamber, 2006).

In spite of the fact that machine learning is its major component, data mining is the confluence of diverse fields such as database technology, statistics, pattern recognition, information retrieval, neural networks, knowledge-based systems, artificial intelligence, high-performance computing, and data visualisation (Han and Kamber, 2006). It can be divided broadly into three active research areas, namely association rule discovery (Agrawal et al., 1993), classification learning (Clark and Niblett, 1989, Quinlan, 1993) and clustering (Jain et al., 1999, Pham and Afify, 2007). Of these, classification learning is the most common data mining activity and is also the focus of this thesis.

Classification learning involves building a model from a set of pre-classified instances referred to as the training dataset which can then be used to classify new instances. It has numerous applications such as in scientific experimentation, manufacturing, telecommunications, medical diagnosis, fraud detection, credit approval and target marketing (Afify, 2004). Some of the examples of classification learning methods used as a part of data mining applications include classifying trends in financial markets and identifying objects in large image databases (Kantardzic, 2003). Different types of algorithms exist for classification learning such as inductive learning algorithms, instance-based algorithms, neural networks, genetic algorithms and bayesian learning algorithms (Afify, 2004). Although the most important consideration in case of classification learning is the success rate of the learned model on test data, the measure of success in many practical data mining applications is more subjective in terms of acceptability of the learned description to a human user (Witten and Frank, 2005). As a result, inductive learning algorithms have become the number one choice for data mining applications since they are not only simple, fast and accurate but are also capable of creating highly structured and comprehensible models.

Inductive learning techniques for the purpose of classification learning can be divided into two main branches, namely decision tree induction and rule induction. A major drawback of both these learning approaches however is their limited capability in handling large datasets. This means that they are not scalable and are consequently unsuitable for data mining applications. Since large datasets are ubiquitous in real-world data mining applications, addressing this problem is of critical importance. Secondly, the existence of noise in these huge data repositories demands that the devised algorithms should also be robust enough in order to be practical.

Early research efforts to address the issue of scalability for decision tree induction algorithms included discretisation of continuous-valued attributes as well as data sampling at each node. More recently, efficient data structures have also been proposed to handle the issue of scalability for decision tree learning (Han and Kamber, 2006). Although decision trees do have their counterparts which can convert the generated tree into a set of rules, rule induction algorithms are often preferred due to several limitations of the decision tree learning paradigm. These include *repetition* and *replication* (Han and Kamber, 2006), *redundancy* (Cendrowska, 1987), and the inability to incorporate domain expert knowledge into the learning phase (Afify, 2004). Furthermore, since post-pruning is always the preferred pruning method for decision trees as compared to pre-pruning, the computational cost of generating rule sets from trees tends to be very high (Witten and Frank, 2005). Apart from this, rule induction algorithms also have the added advantage of their natural extension to the first order inductive logic programming framework (Furnkranz, 1999). However, despite their advantages, little work has been done towards the development of scalable rule induction algorithms for classification learning.

RULES-6 (Pham and Afify, 2005b), developed at the author's laboratory, is a data mining algorithm for classification learning. It employs certain user-specified *search space pruning* techniques and can handle large noisy datasets efficiently as compared to its predecessor RULES-3 Plus. The algorithm however has some limitations, the most prominent of which is an inefficient duplicate rule handling strategy and a suboptimal control structure. The latter was designed to cut down the search space only, although it could be exploited to the fullest by pruning over-specific rules as well. Also, the constraints employed in the control structure are suitable only for mild pruning and therefore result in a minute reduction in the learning time.

The AQ and CN2 family of algorithms remove the covered examples from the training data each time a new rule is induced. This gives rise to problems such as fragmentation and small disjuncts. The RULES family avoids these problems by only marking the examples instead of removing them, which leads to the discovery of globally optimal rules. The drawback of the this approach however is the increased overlapping among rules, which in turn increases the rule set size. Hence, it is critical to address this issue in order to have rule sets that are concise and at the same time avoid the above mentioned problems.

1.2 Research Objectives

The overall objective of this research was to test the hypothesis that the scalability and robustness of RULES-6 can be improved by addressing the identified limitations, by using ideas from association learning, and by developing novel discretisation as well as simple pruning techniques. However, it was also borne in mind that the scalability and robustness achieved should not be at the cost of loss of classification accuracy of the induced model. The algorithms so designed should be able to drastically reduce computational complexity in order to achieve lower execution times, as well as produce smaller rule sets with higher intelligibility.

In this context, the achievement of the research goals outlined above necessitated the following:

- To address the limitations of RULES-6 by using deep rule mining contraints based on ideas from association learning in order to improve its scalability and noise tolerance.

- To develop a new pre-processing discretisation procedure which would improve the generalisation capability of the newly developed algorithm so as to achieve higher classification accuracy.

- To develop new simple pruning techniques which would not only reduce the extent of overlapping between rules but would also enable the algorithm to handle noise in a straightforward manner.

1.3 Research Methodology

To reach the above-mentioned objectives, the following methodology was adopted:

- An in-depth review of existing literature was carried out, describing the state-of-the-art of inductive learning techniques as well as association rule mining techniques. This was intended to uncover the major contributions from the point of view of scalability and robustness for data mining applications.

- A critical assessment of RULES-6 was carried out using a novel manual execution approach. The algorithm improves upon its predecessor RULES-3 plus by introducing certain user-specified pruning techniques that aim to discard portions of the search space that are deemed not to be solutions.

- The newly developed algorithm RULES-7 was compared against its immediate predecessor RULES-6 on a diverse population of datasets using the widely accepted stratified 10-fold cross validation approach. The same approach was used to evaluate the proposed discretisation technique against three other well-known discretisation methods. Finally, RULES-7 with the new pruning techniques was also compared against the default RULES-7 using the above-mentioned evaluation scheme.

1.4 Thesis Outline

This section outlines the organisation of the rest of the thesis.

Chapter 2 scrutinises the existing literature on inductive learning techniques proposed to date, with special emphasis on rule induction algorithms, highlighting any achievements gained in the areas of scalability and robustness. It also reviews well-known scalable association rule mining techniques as well as a new area known as associative classification in order to investigate the applicability of any constraints on the classification learning side.

Chapter 3 starts by highlighting some of the limitations of RULES-6. It then presents a new rule induction algorithm RULES-7, which improves upon its immediate predecessor by utilising three new techniques, two of which have been adapted from association rule mining. The efficacy of the new techniques is discussed in detail and illustrated by means of examples. The chapter concludes with an empirical evaluation of RULES-7 against RULES-6 to demonstrate the improvements achieved with respect to performance and accuracy.

Chapter 4 begins with a review of major discretisation approaches proposed to date in the context of inductive learning. It then presents a new pre-processing discretisation technique EDISC based on Entropy-MDLP, a discretisation method that has been accepted as by far the most effective in terms of classification accuracy. The innovation in the new technique however is that it takes a class-centered approach to discretisation, which results in the discovery of optimal cut points for each class present in an attribute. Empirical evaluation of the new technique against three other state-of-the-art discretisation methods presented at the end of the chapter demonstrates its superiority.

Chapter 5 reviews current pre-pruning, post-pruning, and hybrid-pruning techniques proposed for decision tree induction and rule induction. It then proposes a new hybrid pruning technique for RULES-7 aimed at addressing the issue of overlapping inherent to the RULES family. This is followed by suggesting a second incremental post-pruning technique which utilises a misclassification tolerance in order to cope with noisy data. Experimental evaluation proves that the former technique considerably reduces the extent of overlapping for RULES-7, whereas the latter augments its noise handling capability, resulting in a rule set that is not only compact but also more accurate.

Chapter 6 concludes the thesis by summarising its major contributions and suggesting directions for future research.

CHAPTER 2

SURVEY OF CLASSIFICATION LEARNING AND RELATED FIELDS

2.1 Foreword

Classification learning is a type of *supervised learning* in which the class labels of the instances as well as the total number of classes are known beforehand. The system has to induce a class description (aka concept description), given instances belonging to that class referred to as positive instances and those not belonging to it referred to as negative instances. This is in contrast to *clustering* which is *unsupervised* and in which the task is to group the instances into new categories based on some similarity. Classification learning also has a strong linkage with association rule mining, which differs in that it is not restricted to predicting a single attribute but can also predict combinations of attributes. Another difference is that as opposed to classification rules that are meant to be used as a set, each association rule is used individually in order to express the different regularities that underlie the dataset (Witten and Frank, 2005).

An important characteristic of real-world datasets is that they inevitably contain some degree of noise. This means that the attribute values for certain examples in the dataset may not only be missing but there might also be inconsistencies in attribute values. As already mentioned in chapter 1, data for machine learning applications is limited in quantity, which makes it extremely easy to remove such noise from it prior to learning. As a result, each example is complete and the values of all its attributes are available during learning, which leaves no ambiguity in determining its class (Afify, 2004). The domain of data mining is real-world data stored in gigantic databases, which is also imperfect. However, due to its enormously large size, it is not so easy to eliminate noise from such data and so the learning algorithms must be robust enough to cope with such large and noisy datasets.

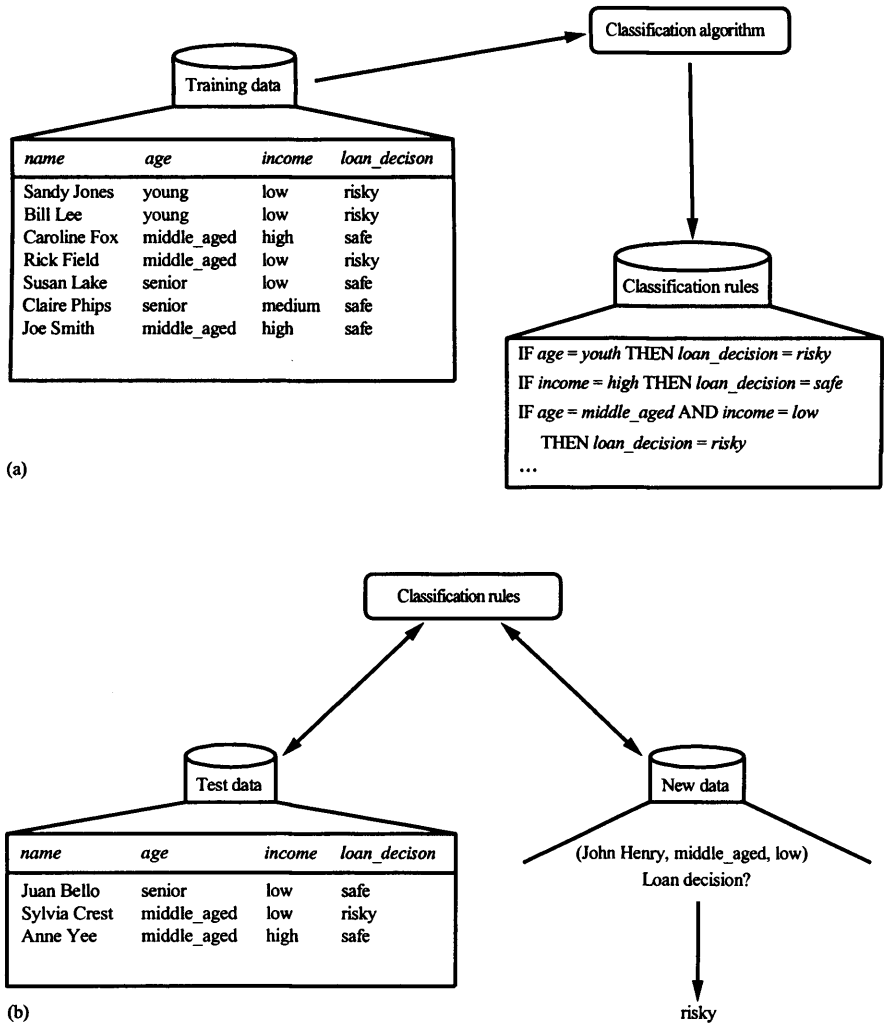
Although there are many approaches for classification learning other than inductive learning, such as instance-based algorithms, neural networks, genetic algorithms and bayesian learning algorithms, almost all of them have one or more limitations which renders them unsuitable for data mining applications. For example instance-based learners are known to have poor noise immunity and to have large memory requirements because of their inability to decide which instances to store for use during generalisation, resulting in a slow execution speed (Wilson and Martinez, 2000). Other drawbacks of instance-based learners that require investigation include the derivation of concise abstractions, the learning of data structures for indexing, as well as similarity definitions for nominal-valued attributes (Aha et al., 1991). Although neural networks do have a high tolerance for noisy data, they also have their limitations such as long training times, the requirement of a number of parameters that are typically best determined empirically, as well as poor interpretability (Han and Kamber, 2006). Similarly, genetic algorithms have high execution times, suffer from randomness in creating the initial population, and are susceptible to myopia after finding a single good solution (Dhar et al., 2000). Finally, the limitations of bayesian networks include a lack of expressiveness, a strong dependence on prior knowledge, computational complexity, and the inability to handle the unanticipated probability of an event (Niedermayer, 2008).

In view of the reasons outlined above, this chapter will only focus on and elaborate the inductive learning approaches to classification along with investigating the closely related area of association learning. The chapter is organised as follows. Section 2.2 presents the fundamentals of classification learning paradigms, including input and output format as well as model evaluation. Section 2.3 describes in detail inductive learning approaches to classsification including major decision trees and rule induction algorithms. It also highlights their contribution on scalability issues. Section 2.4 discusses the essentials of association learning paradigms such as frequent itemset mining, generation of association rules, as well as rule mining constraints to limit the search for interesting rules. Section 2.5 delineates a new approach to classification learning known as associative classification, along with the description of some representative algorithms that exploit the new methodology. A summary of the chapter is presented in section 2.6.

2.2 Fundamentals of Classification Learning

Data classification is essentially a two-step process as outlined in Figure 2.1 (Han and Kamber, 2006). In the first step, the classification learning system is given as input a set of examples with known class labels that have been selected from the database under analysis. This set is referred to as the training dataset and the goal of the learner in the context of inductive learning is to induce from the data a classification model, also known as a classifier, theory or hypothesis, which comprises a description for each class. In other words, the output of learning is a classification model analogous to a mapping function that can map an example in the training data into one of several predefined classes, given the values of its other attributes (Kantardzic, 2003).

The attribute describing the class label of the example is called the dependent attribute, whereas the other attributes are often called independent or predictor attributes (Gehrke, 2006). The accuracy with which the induced model can predict the class labels of examples in



name	*age*	*income*	*loan_decison*
Sandy Jones	young	low	risky
Bill Lee	young	low	risky
Caroline Fox	middle_aged	high	safe
Rick Field	middle_aged	low	risky
Susan Lake	senior	low	safe
Claire Phips	senior	medium	safe
Joe Smith	middle_aged	high	safe

IF *age* = *youth* THEN *loan_decision* = *risky*
IF *income* = *high* THEN *loan_decision* = *safe*
IF *age* = *middle_aged* AND *income* = *low*
THEN *loan_decision* = *risky*
...

name	*age*	*income*	*loan_decison*
Juan Bello	senior	low	safe
Sylvia Crest	middle_aged	low	risky
Anne Yee	middle_aged	high	safe

Figure 2.1 The Data Classification Process. (a) Learning Phase. (b) Classification Phase [Han and Kamber, 2006].

the training data is referred to as the training accuracy. This is defined as the percentage of examples in the training data whose class label is the same as described by the model. The learned classification model can be expressed in one of two output structures for the purpose of inductive learning, namely a decision tree or a rule set. The structure depicted in Figure 2.1(a) is a rule set that can be used to classify future loan applications as being either safe or risky. It can also be used to provide a deeper insight into the database contents, as well as to express them in a more compressed form.

In the second step, the learned model is deployed for predicting the class labels of future examples, which is the most important objective in case of classification learning. However, it is critical to estimate the true accuracy of the classifier before doing this owing to the tendency of the classifier to overfit the training data, which means that it may follow the particular details of the data too slavishly (Witten and Frank, 2005). This phenomenon of overfitting is more prominent in the presence of noise in the training data, hence it is important to evaluate the classifier on a separate set of examples that have not been used during training. These examples are chosen randomly from the database under analysis and are referred to as the test dataset. The class labels of examples in the test data are compared with those predicted by the model, which gives its *classification accuracy.* This is defined as the percentage of examples in the test data whose class label is the same as predicted by the model. If the classification accuracy is considered acceptable, the model can be used to classify future examples for which the class label is unknown. For instance, the rule set shown in Figure 2.1 can be used to approve or reject new loan applications.

If the attribute to be predicted is discrete-valued and unordered (categorical), the learning task is termed classification as outlined above. The term categorical implies that each value of the attribute defines a category or class. On the other hand, if the attribute is continuous-valued and ordered (numerical), then the task becomes that of *prediction,* in which the goal is to predict the loan amount in dollars that can be lent safely to an applicant. This thesis focuses only on classification learning.

2.2.1 Format of Training Data

The format of input data is more or less the same for the majority of inductive learning algorithms where the examples correspond to rows in a table and the attributes correspond to columns. In the machine learning literature, an example is also referred to as an object, record, tuple, etc. whereas an attribute is commonly referred to as a feature or field. Since an example E can have only one value for a particular attribute A, it can simply be termed as "a collection of attribute values". More formally, an example E can be represented by an n-dimensional attribute vector:

$$E \rightarrow (A_1 = v_1^E,\ A_2 = v_2^E, \dots,\ A_n = v_n^E)$$

Where $A_1, A_2, \dots, A_n$ are the attributes present in the database and $v_1^E, v_2^E, \dots, v_n^E$ are the values of these attributes for the example E. This is illustrated in Table 2.1 using the contact lens data (Witten and Frank, 2005).

An attribute can either be categorical (discrete) or continuous (numerical). If the attribute is categorical and unordered such as "Astigmatism" in Table 2.1, it is referred to as a *nominal* attribute. If the attribute is categorical and ordered with an implied ranking among the values such as "Recommended Lenses", it is termed an *ordinal* attribute. The contact lens data shown in Table 2.1 is an example of categorical type data. However, since the terms categorical and nominal are used almost synonymously in the machine learning literature, such datasets will be referred to as nominal in this thesis. A continuous attribute on the other

Attributes → **Examples ↓**	A_1 = **Age**	A_2 = **Spectacle prescription**	A_3 = **Astigmatism**	A_4 = **Tear production rate**	A_5 = **Recommended lenses**
E_1 =	young = v_1^E	myope = v_2^E	no = v_3^E	reduced = v_4^E	none = v_5^E
E_2 =	young	myope	no	normal	soft
E_3 =	young	myope	yes	reduced	none
E_4 =	young	myope	yes	normal	hard
E_5 =	young	hypermetrope	no	reduced	none
E_6 =	young	hypermetrope	no	normal	soft
E_7 =	young	hypermetrope	yes	reduced	none
E_8 =	young	hypermetrope	yes	normal	hard
E_9 =	pre-presbyopic	myope	no	reduced	none
E_{10} =	pre-presbyopic	myope	no	normal	soft
E_{11} =	pre-presbyopic	myope	yes	reduced	none
E_{12} =	pre-presbyopic	myope	yes	normal	hard
E_{13} =	pre-presbyopic	hypermetrope	no	reduced	none
E_{14} =	pre-presbyopic	hypermetrope	no	normal	soft
E_{15} =	pre-presbyopic	hypermetrope	yes	reduced	none
E_{16} =	pre-presbyopic	hypermetrope	yes	normal	none
E_{17} =	presbyopic	myope	no	reduced	none
E_{18} =	presbyopic	myope	no	normal	none
E_{19} =	presbyopic	myope	yes	reduced	none
E_{20} =	presbyopic	myope	yes	normal	hard
E_{21} =	presbyopic	hypermetrope	no	reduced	none
E_{22} =	presbyopic	hypermetrope	no	normal	soft
E_{23} =	presbyopic	hypermetrope	yes	reduced	none
E_{24} =	presbyopic	hypermetrope	yes	normal	none

Table 2.1 The Contact Lens Data (nominal-type) [Witten and Frank, 2005].

hand does not assume discrete labels as its values but instead a range of numeric values. A great majority of real-world datasets are either of continuous type in which all the attributes in the dataset are continuous, or of mixed type in which some of the attributes are continuous whereas the rest are nominal. Examples of continuous and mixed type datasets are shown in Table 2.2 and Table 2.3 respectively (Witten and Frank, 2005).

For the purpose of classification learning, any nominal attribute in the database can be chosen as the class attribute (the attribute on whose values the examples are to be classified). In the case mentioned above, if the n_{th} attribute is nominal and is chosen as the class attribute then,

$$E \rightarrow (A_1 = v_1^E,\ A_2 = v_2^E, \ldots, \text{Class} = v_n^E)$$

2.2.2 Model Representation

As already mentioned, the classification model in case of inductive learning can be expressed either as a decision tree or as a rule set. Both the structures have the advantage that they are highly comprehensible. Furthermore, both follow a general to specific search methodology (Mitchell, 1997) which favors general hypotheses over more complex ones, consequently biasing the learning system towards generality.

2.2.2.1 Decision Tree Induction

A decision tree for the contact lens data of table 2.1 is shown in Figure 2.2. As can be seen from the figure, a decision tree consists of internal nodes and leaf nodes. The internal nodes are represented by ovals in the figure whereas the leaf nodes are indicated by rectangles with gray background. The topmost node titled “tear production rate” in the tree in Figure 2.2 is referred to as the root node and represents the most general hypothesis. Each internal node corresponds to a test on an attribute whereas the branches from that node correspond to the

	Sepal length (cm)	Sepal width (cm)	Petal length (cm)	Petal width (cm)	Type
1	5.1	3.5	1.4	0.2	*Iris_setosa*
2	4.9	3.0	1.4	0.2	*Iris_setosa*
3	4.7	3.2	1.3	0.2	*Iris_setosa*
4	4.6	3.1	1.5	0.2	*Iris_setosa*
5	5.0	3.6	1.4	0.2	*Iris_setosa*
...					
51	7.0	3.2	4.7	1.4	*Iris_versicolor*
52	6.4	3.2	4.5	1.5	*Iris_versicolor*
53	6.9	3.1	4.9	1.5	*Iris_versicolor*
54	5.5	2.3	4.0	1.3	*Iris_versicolor*
55	6.5	2.8	4.6	1.5	*Iris_versicolor*
...					
101	6.3	3.3	6.0	2.5	*Iris_virginica*
102	5.8	2.7	5.1	1.9	*Iris_virginica*
103	7.1	3.0	5.9	2.1	*Iris_virginica*
104	6.3	2.9	5.6	1.8	*Iris_virginica*
105	6.5	3.0	5.8	2.2	*Iris_virginica*
...					

Table 2.2 The Iris Data (continuous-type) [Witten and Frank, 2005].

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

Table 2.3 The Weather Data (mixed-type) [Witten and Frank, 2005].

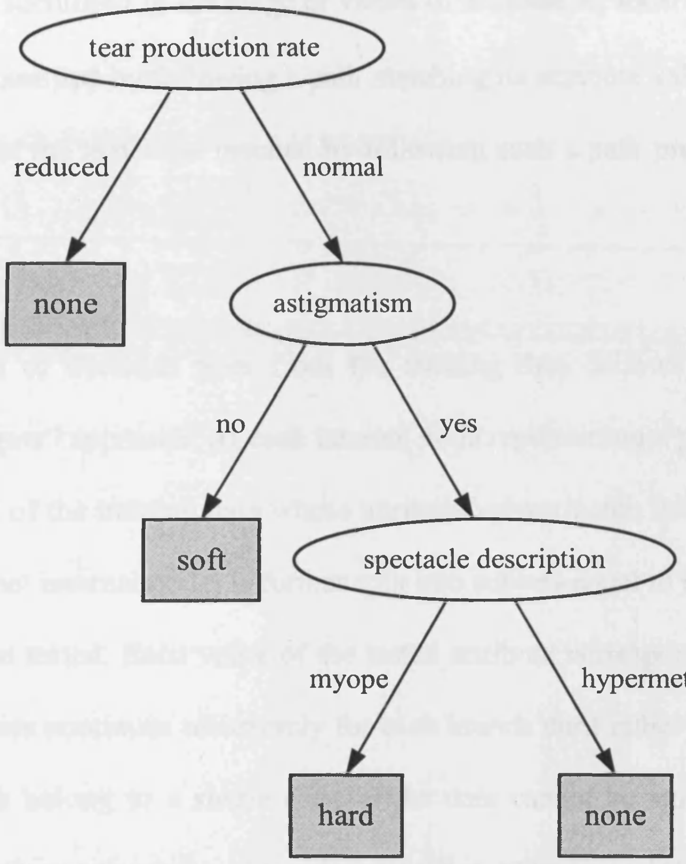


Figure 2.2 Decision Tree for the Contact Lens Data [Witten and Frank, 2005].

result of that test. The number of branches for a nominal attribute is equal to the number of values of that attribute. In the case of a continuous attribute, the number of branches may either be two if binary discretisation is adopted or equal to the number of splits identified in the case that multi-interval discretisation is chosen. For the simple case of binary discretisation, each branch corresponds to one of the two tests $A_i \leq t_{ij}$ and $A_i > t_{ij}$, where t_{ij} is a threshold identified in the range of values of attribute A_i local to that particular node. An example is classified by following a path matching its attribute values from the root node of the tree, so that the leaf node reached by following such a path predicts the class label of the example.

The construction of decision trees from the training data follows a recursive top-down "divide-and-conquer" approach. At each internal node representing a particular attribute to be tested, the subset of the training data whose attribute values match those of the path from the root node up to that internal node, is further split into subsets equal to the number of values of the attribute to be tested. Each value of the tested attribute corresponds to a branch and the partitioning process continues recursively for each branch until either all the examples going down that branch belong to a single class or the data cannot be split any further. At each recursion, an *attribute selection measure* is required in order to decide on the best attribute to split, which is the point of major difference among the different decision tree induction algorithms. In the presence of noise or outliers, a decision tree so created may be vulnerable to overfitting of the training data. Consequently, a tree simplification procedure, referred to as pruning, is required. Pruning is the topic of chapter 5 and is another cause of divergence among the various decision tree algorithms.

Decision trees are popular for several reasons, such as their ability to perform exploratory knowledge discovery without requiring any domain knowledge or parameter setting, ability to handle high dimensional data, as well as because of their simplicity, speed and accuracy. They can easily be converted into a set of rules with each rule corresponding to a path from the root node to a leaf, resulting in an unordered rule set with mutually exclusive rules. However, they suffer from many problems which do not favor the rule extraction approach from a decision tree. For example, they are prone to *repetition* and *replication* which can adversely affect their interpretability (Han and Kamber, 2006). Repetition can be caused by a continuous attribute being tested repeatedly along a given tree branch, whereas replication occurs due to the existence of duplicate subtrees within the tree. These two problems exist only in case of *univariate* decision trees, which split on a single attribute at a time. They can be prevented by using *multivariate* decision trees which operate by splitting on a combination of attributes and so are more accurate and smaller than their univariate counterparts. However, the generation of multivariate decision trees is not only computationally expensive but they are also more difficult to interpret (Witten and Frank, 2005). Decision trees also suffer from redundancy (Cendrowska, 1987), which means that they are unable to discern attribute relevancy because of their emphasis on minimising the average entropy of the training data. Consequently, the extracted rule set requires a comprehensive pruning phase in order to remove the redundant conditions, because of which rule generation from trees tends to be extremely slow. In view of the above mentioned plethora of problems, it was critical to come up with an alternative approach in order to generate the rule sets directly from the training data, which in turn led to the development of rule induction algorithms.

2.2.2.2 Rule Induction

A rule induction algorithm is capable of generating IF-THEN rules directly from the training data. The format of each rule is:

IF *Condition$_1$* AND *Condition$_2$* AND ... *Condition$_i$* THEN *Class$_j$*

The conditions occurring between the IF and THEN parts of the rule are collectively referred to as its *antecedent*. Each condition corresponds to either an attribute-value pair $[A_i = v_{ij}]$ in the case of a nominal attribute or the attribute bounded by upper and lower threshold values in its range $[t_{i1} < A_i \leq t_{i2}]$ in the case of a continuous attribute. The number of examples in the dataset satisfying the antecedent of the rule is referred to as its *coverage*. The single condition occurring after the THEN part represents the target class of the rule and is referred to as its *consequent*. It corresponds to the attribute-value pair $A_j = v_{jk}$, where A_j is the attribute chosen as the class. The *consistency* or *accuracy* of the rule is defined as the number of examples satisfying both its antecedent and consequent divided by the number of examples satisfying only the antecedent.

A rule induction algorithm is also referred to as a *sequential covering algorithm* since it operates sequentially by learning one rule at a time. The induced rule aims to cover as many examples of the target class as possible while excluding examples belonging to other classes. In the machine learning literature, examples of the target class are referred to as positive examples whereas those belonging to other classes are referred to as negative examples. Rule induction follows a "separate-and-conquer" approach in which the examples covered by a newly induced rule are separated and the induction continues on the remaining examples. In this regard, it is different from decision tree induction which is analogous to learning all the rules simultaneously, as induction proceeds downward from the root node to the leaves (Witten and Frank, 2005).

Rule induction begins with a rule with empty antecedent, i.e. IF ___ THEN $Class_j$ covering all the examples in the dataset. This most general rule is specialised gradually at each iteration by adding a single condition to its antecedent, and the quality of the resulting rule is assessed using a *rule evaluation measure*. The rule evaluation measure is also referred to as the *specialisation heuristic* and varies from one rule induction algorithm to another.

Another important component of rule induction is the *search strategy* used for arriving at the best rule from the most general rule. Since the number of attributes in the training dataset as well as the number of values each attribute can take may be extremely large, evaluating every possible combination of attribute values becomes a combinatorial problem. This is referred to as the *exhaustive search* strategy which aims to discover the optimal rule set. At the other end of the spectrum, the *greedy search* strategy retains only a single rule with the highest quality measure at the end of every specialisation iteration. However, it is susceptible to myopia (Witten and Frank, 2005) since a poorly made choice cannot be undone during the later stages of rule induction. To avoid the computational explosion associated with exhaustive search and the myopia associated with greedy search, an alternative strategy referred to as *beam search* is commonly used. Beam search retains a set of alternative rules at the end of each specialisation iteration whose number is equal to the beam width w. The chances of myopia can be reduced significantly with a reasonable beam width, although it may still occur if the beam width is not large enough.

Rule induction algorithms can either generate an ordered set of rules, referred to as a decision list (Rivest, 1987), or an unordered set of rules. In case of a decision list, the rules are meant to be executed in order which means that a rule appearing earlier in the rule set has the highest priority to classify an example than the one that comes afterwards. This is referred to as a *rule-based* ordering scheme, which may utilise either the rule's accuracy, coverage, the number of attributes in its antecedent or domain expert advice, in order to rank the rules. The alternative is *class-based* ordering, in which rules for the most frequent class appear first in the list whereas those for the least frequent class appear at the end (Han and Kamber, 2006). This scheme may also adopt the misclassification cost per class for class ordering. The rules within a class, however, are not ordered since all of them predict the same class.

In the case of unordered rule sets, there is no precedence among the rules, which means that multiple rules may be fired in case their antecedent is satisfied by an example. In this scenario, either a conflict resolution approach may be required to decide the best rule to classify an example, or a weighted voting scheme may be chosen, in which multiple rules contribute towards the classification decision. Unordered rule sets have the advantage that each rule represents a standalone piece of knowledge and can be removed or modified without disturbing the rest of the rule set, which makes them more flexible so that they can be easily managed by human experts. Their only disadvantage is that the overall number of rules needed to capture the concept increases because of rules covering common areas within the instance space.

2.2.3 Model Evaluation

Different evaluation schemes exist in order to estimate the worth of the induced model objectively in terms of classification accuracy. Some of the most common ones include the *holdout method*, *cross validation* and *bootstrap*. The holdout method is the simplest of these, which randomly partitions the dataset into a training set and a test set, usually reserving two thirds for training and one third for testing. The estimate of classification accuracy using this method is pessimistic because of not using a certain part of data during training. A variation of this method is *random subsampling,* in which the holdout method is repeated k times so that the classification accuracy is the average of the k iterations.

Cross validation is another evaluation scheme which randomly partitions the dataset into k mutually exclusive subsets or folds. The model is trained using $k - 1$ folds and tested on the remaining one fold. If this procedure is repeated n times, it is referred to as *n-fold cross validation.* Furthermore, it is also important to carry out stratification so as to ensure that the class distribution of examples in the training and test data is the same as that in the original dataset, in which case this scheme is termed *stratified n-fold cross validation.* It has been found empirically that setting the number of folds k equal to 10 gives the most reliable accuracy estimate for the majority of datasets because of its relatively low bias and variance (Kohavi, 1995b). Despite the ongoing debate within the data mining community as to what should be the right number of folds, stratified 10-fold cross validation has become the standard method of evaluating the performance of any learning technique. A variation of cross validation is the *leave-one-out* method, where k is set equal to the total number of examples in the dataset and training is performed a total of k times each time leaving out only one example for the test set. However, this scheme is not only computationally expensive for large datasets but can also lead to an estimated error rate of as much as 100%.

Bootstrap is yet another evaluation method that is very similar to the ones outlined above, with the exception that an example may be selected twice for the training set, which is referred to as *sampling with replacement*. Its operation is based on the fact that many learning techniques can use the same example twice, thereby making a difference in the result of learning (Witten and Frank, 2005). The most common bootstrap method is the .632 bootstrap. It derives its name from probability theory, which shows that for a given dataset of *n* examples which is sampled *n* times with replacement, 63.2% of the original examples will end up in the bootstrap whereas 36.8% will form the test set. However, because of sampling with replacement, the number of examples in the training set remains equal to *n,* which means that 36.8 % of examples in it have been selected more than once from the original dataset.

2.3 Major Inductive Learning Algorithms for Classification

A panoply of inductive learning algorithms for classification have been proposed to date, based on both the decision tree and the rule induction learning paradigm. However, the fundamental approach is more or less the same as the one proposed in some of the primitive learning algorithms. This section therefore discusses only the major decision tree and rule induction algorithms.

2.3.1 Decision Tree Algorithms

One of the earliest decision tree induction algorithms is ID3 (Quinlan, 1983, Quinlan, 1986), which stands for *Iterative Dichotomiser.* It uses *information gain* as the attribute selection measure, which has its roots in information theory (Shannon and Weaver, 1963). Finding the information gain involves calculating the *entropy* of the data reaching a particular node, and subtracting from it the entropy of the data partitions created by splitting the node on a particular attribute. The entropy of any dataset *D* containing *k* classes is given by:

$$Entropy(D) = -\sum_{i=1}^{k} \frac{|C_i|}{|D|} log_2 \frac{|C_i|}{|D|} \qquad (2.1)$$

where $|C_i|$ is the number of examples of class i in the dataset and $|D|$ is the total number of examples in the dataset. The entropy of the dataset represented by equation 2.1 is synonymous with the *information required* to classify the examples in D without partitioning on any attribute.

Entropy in this context is a measure of the amount of disorder within a dataset D. A dataset will have zero disorder if and only if there is a single class in it, in which case no partitioning will be required. Whenever there is more than one class, there will be some degree of disorder and so it will be necessary to split on some attribute, which results in the least disorder thereby producing the purest possible partitions. If an attribute A is discrete-valued and has n possible values $\{a_1, a_2, \dots, a_n\}$, then n partitions will be created in the dataset D by splitting on this attribute, i.e. $\{D_1, D_2, \dots, D_n\}$. The entropy of the attribute A will therefore be given by:

$$Entropy_A(D) = -\sum_{j=1}^{n} \frac{|D_j|}{|D|} \times Entropy(D_j) \qquad (2.2)$$

where $|D_j|$ is the number of examples going down the j^{th} branch. In the case when the attribute is continuous-valued, a binary partition is created by identifying a split point or threshold on the values of A. This is accomplished by sorting its values in ascending order and placing a potential split point in the middle of two consecutive unique values. Each potential split point is then evaluated by substituting $n = 2$ in equation 2.2. The split point with the lowest entropy is selected as the threshold, which creates two branches on attribute A, $A \leq split\_point$ and $A > split\_point$. The entropy of attribute A represented by equation 2.2 is also synonymous with the information required to classify the examples by splitting on the attribute A. The lower this value the lesser information will be required to classify the examples because of the higher purity of the resulting partitions.

Information gain is the difference between the entropy (information required) before partitioning and after partitioning on a particular attribute. It is defined as:

$$Gain(A) = Entropy(D) - Entropy_A(D) \tag{2.3}$$

As the name suggests, information gain is an expression of how much has been gained by splitting on the attribute A and is analogous to the expected reduction in the information requirement because of knowing the value of A (Han and Kamber, 2006). Entropy and information gain of an attribute express the same thing at the opposite ends of the spectrum. The lower the entropy resulting from partitioning on a particular attribute and the higher the information gain, the better the attribute will perform at classifying the examples within the dataset.

One problem with the information gain criterion is that it is biased towards attributes with many values because of its emphasis on creating pure partitions. The limiting case can be an *index* attribute having n numbers as its values, one for each example in the dataset. Splitting on this attribute will result in n branches, each having a single example in it belonging to some class. Since the resulting partitions are pure, the information gained by splitting on the index attribute will be maximum and so it will be selected over other attributes. However, this results in no generalisation and is consequently worthless for classification learning.

In order to address this issue, an extension of ID3 referred to as C4.5 was proposed (Quinlan, 1993) which employed the *gain ratio* criterion, a modified version of information gain defined as:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)} \tag{2.4}$$

where $SplitInfo(A)$ is analogous to equation 2.2 and is defined as:

$$SplitInfo(A) = -\sum^{n} \frac{|D_j|}{|D|} log_2 \frac{|D_j|}{|D|}$$

$$SplitInfo(A) = -\sum_{j=1} \frac{|D_j|}{|D|} log_2 \frac{|D_j|}{|D|} \tag{2.5}$$

However, it is different from equation 2.2 in that it does not measure the amount of information required for classification by splitting on the attribute A, but instead the number of examples going down each branch out of the total number of examples in the dataset D. This is to counter the bias towards highly branching attributes by ensuring that each branch contains a sufficient number of examples. The gain ratio is still prone to instability in situations where the *SplitInfo* approaches zero. However, this can easily be avoided by imposing a constraint that the information gain of the test selected must be at least as high as the average information gain of all the tests.

C4.5 improves upon its predecessor ID3 in many aspects, the most significant of which is the introduction of a post-pruning strategy. The generated complete decision tree is inspected in a bottom-up direction and the error rate is estimated at any parent node, i.e. a node with a subtree. Subsequently, an attempt is made to remove the subtree at the node so that it becomes a leaf node labeled with the majority class of the examples going down that branch and the error rate is recalculated. If the error rate after removal of the subtree is lower, then the subtree is pruned, otherwise it is kept. The error rate is estimated on the training dataset instead of a pruning set and is therefore highly optimistic, since the generated tree is overfitted to the training data. To counter this, a pessimistic penalty function is used to increase the error estimate and consequently this post-pruning strategy is referred to as *pessimistic pruning*. The most recent version of ID3 is C5.0 (RuleQuest, 2001), a commercial product that is not only several orders of magnitude faster than C4.5 but is also more efficient in its use of memory.

CART (Breiman et al., 1984), which stands for *Classification and Regression Trees,* is another decision tree induction algorithm which was developed around the same time as ID3. It uses the *Gini index* as the splitting criterion, which is a measure of impurity in the dataset D and is defined as:

$$Gini(D) = 1 - \sum_{i=1}^{k} \left(\frac{|C_j|}{|D|} \right)^2 \qquad (2.6)$$

CART differs from ID3 in that a node is split into exactly two branches even if the attribute is discrete-valued, resulting in the generation of a binary tree. Before determining the best splitting attribute, it is therefore necessary to evaluate all possible subsets of the values of that attribute in order to arrive at the *best splitting subset.* For example, if the attribute A has 3 values {sunny, rainy, overcast}, then the possible subsets are {}, {sunny}, {rainy}, {overcast}, {sunny, rainy}, {sunny, overcast}, {rainy, overcast} and {sunny, rainy, overcast}. Excluding the power set and the empty set, it means that for an attribute with n values, there are $2^n - 2$ subsets that need to be evaluated. For a given binary split, the Gini index of the attribute A is calculated as follows:

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \qquad (2.7)$$

Each potential binary split for the attribute is evaluated using equation 2.7 and the subset corresponding to the split with the minimum Gini index is chosen as the splitting subset. For a continuous attribute the procedure is similar to that of information gain, which results in the identification of the best split point for which two branches are created on A, $A \leq split\_point$ and $A > split\_point$. Analogous to the information gain in ID3, the impurity reduction in case of CART is defined as:

$$\Delta Gini(A) = Gini(D) - Gini_A(D) \qquad (2.8)$$

The attribute with the maximum impurity reduction is chosen as the splitting attribute. After the complete tree has been generated, it is pruned using a post-pruning strategy similar to the one used in C4.5. However, it employs a separate pruning set for estimating the error rate. This is referred to as *cost complexity* pruning.

2.3.2 Rule Induction Algorithms

AQ (Michalski, 1969), which stands for *Algorithm Quasi-optimal* is one of the first rule induction algorithms. Its most well-known and popular version is AQ15 (Michalski et al., 1986) and the most recent one is AQ21 (Wojtusiak et al., 2006). It represents the learned rules in the form of expressions using the *Variable-valued Logic* system 1 (VL_1), a multiple-valued logic propositional calculus with typed variables (Michalski and Larson, 1975). A condition is referred to as a *selector* in VL_1 which is different in that an attribute in a selector is not related to only a single value but can have multiple values separated by a logical disjunction. Furthermore, any of the relational operators $=$, $\neq$, $>$, $\geq$, $<$, $\leq$ may be used within a selector. A rule is referred to as a *complex* in AQ15 which is a conjunction of selectors, whereas the rule set is referred to as a *cover* which is the disjunction of complexes (Grzymala-Busse and Shah, 2000). The evaluation measure used in AQ15 is the lexicographic functional LEF (Hong et al., 1986) which comprises several criterion-tolerance pairs whose ordering determines the relative importance of each. The tolerance specifies the allowable error within each criterion. A distinctive feature of AQ15 is its ability to perform constructive induction in which it uses its background knowledge to construct new attributes which were not present in the original training data. Such background knowledge is expressed as rules of two different types, *L-type* rules and *A-type* rules which can create new attributes logically and arithmetically from existing ones respectively.

The AQ15 algorithm uses a beam search strategy to search through the hypothesis space by implementing the STAR method of inductive learning (Michalski and Larson, 1975) in which the size of the star corresponds to the beam width specified by the user. The algorithm takes two arguments POS and NEG, describing the set of examples belonging to the target class and those not belonging to it respectively. It works on a class per class basis, selecting a seed example from the target class every time a rule is to be induced. Subsequently, the STAR procedure is called with the arguments SEED and NEG and the specialisation phase starts. This continues as long as any rule in STAR has negative coverage, in which case an attempt is made to exclude from the rule's coverage a negative example E_{neg} closest to the SEED. When all the rules in STAR have become consistent (cover no negative examples), those subsumed by other rules are removed from it. This is followed by the removal of the worst rules from STAR until its size becomes equal to the user-specified beam width. The STAR is then returned to the main algorithm, which adds the best rule in it as an extra disjunct to the rule set for the target class.

The AQ15 algorithm searches only the space of rules that are completely consistent with the training data. As a result, it is prone to overfitting in the presence of noise and so a post-pruning technique is employed in order to remove redundant conditions from rules and redundant rules from the rule set. A more efficient pruning strategy was adopted in a later version of the algorithm referred to as POSEIDON or AQ16 (Bergadano et al., 1992). POSEIDON can not only prune conditions and rules but can also extend and contract the intervals in case of continuous conditions. It continues pruning as long as the coverage, as well as the quality, of the resulting rule set increases.

The AQ15 algorithm can generate ordered as well as unordered rule sets. In the case of an ordered rule set, there is no ambiguity, since the target class of the first rule in the list is assigned to an example that satisfies it. In the case that no rule is satisfied by the example, it is assigned the class of the default rule. To classify an example in the case of unordered rules, the AQ15 algorithm uses two strategies, namely a *strict match* and an *analogical match*. In the case of strict match, a rule has to be satisfied by an example exactly. The analogical match by contrast determines the closeness between the example and the rule. An interesting situation in which the analogical match can be extremely useful is the one in which each rule is associated with a pair of weights *t* and *u*, representing the total number of examples and the unique number of examples classified by the rule respectively. The higher the *t*-weight, the more representative the rule is considered since it describes the most typical examples of the training population. On the other hand, a rule with the lowest *u*-weight can be viewed as describing rare exceptional examples. In the presence of noise, such rules can be removed successively, starting with the one having the lowest *u*-weight. This method of knowledge reduction by truncation is referred to as TRUNC. In contrast to the exact match strategy, the analogical match in this case can still classify examples by assessing their closeness to the correct concept.

When the strict match strategy is used, there are again three possibilities. The first is the one in which only one rule is satisfied by a particular example, in which case its class is predicted to be the target class of the rule. In the case that the example satisfies more than one rule, an *estimate of probability (EP)* is calculated for each rule. This is equal to the number of examples covered by it out of the total number of examples in the training data, in case the rule is satisfied by the example and zero otherwise. The estimate of probability EP of a class is then calculated as the probabilistic sum of the EPs of the rules of that class. The class with the highest EP is taken to be the class of the example. The last case in which none of the rules is satisfied by the example is dealt with by evaluating a *measure of fit (MF)* for each class. This involves calculating the MF of each condition within a rule. The MF of the condition is equal to 1 if it is satisfied by the example. If this is not the case, then it is equal to the number of values that an attribute is assuming within the condition disjunctively divided by the total number of values within the attribute's domain. The MF of the complex is then defined as the product of the MF of each of its constituent conditions multiplied by the ratio of the number of examples covered by the rule to the total number of examples within the training data. Finally, the measure of fit MF of a class is calculated as the probabilistic sum of the MFs of the rules of that class.

CN2 (Clark and Niblett, 1989) is another rule induction algorithm named after its authors. It attempts to combine the good features of the decision tree algorithm ID3 with the rule induction algorithm AQ. CN2 uses a subset of the expression language VL_1 used in AQ and also retains the beam search strategy of AQ. However, it differs from AQ in that it does not rely on specific examples during search but instead considers all specialisations of a complex similar to ID3, which considers all attributes in order to find the best one to split at a particular node. The specialisation of a complex involves either adding a new conjunctive term or removing a disjunctive element in one of its selectors. Because of this top-down search for complexes, the CN2 algorithm is able to incorporate a cutoff method similar to the one used in decision tree pruning that can halt specialisation of a complex when no further statistically significant specialisations can be found. This results in an extension of the search space to include rules that do not perform perfectly on the training data.

The CN2 algorithm handles continuous attributes by dividing the range of values of each attribute into discrete subranges and then creating two thresholds on the attribute at subrange boundaries similar to ID3. It also takes care of any missing values for both discrete and continuous-valued attributes. In case of discrete attributes, the missing value is replaced with the most commonly occurring value of that attribute in the training data. For continuous attributes, the mid-value of the most commonly occurring subrange is used to replace any missing values.

The original version of CN2 produces an ordered set of rules. In the case where an example satisfies none of the rules, it is classified using the default rule at the end of the list, which assigns it the class label of the most frequently occurring class within the training data. CN2 uses two rule evaluation measures, namely entropy and *statistical significance*, the latter being measured using the likelihood ratio statistic (Kalbfleish, 1979) which is defined as:

$$2\sum_{i=1}^{n} f_i \, log\left({f_i}/{e_i}\right) \tag{2.9}$$

The function assesses the significance of the complex by comparing the observed frequency f_i of examples satisfying the complex among classes with the expected frequency e_i if the rule made random predictions. The higher this ratio, the greater the likelihood that the performance of the rule is not due to mere chance.

The CN2 algorithm was modified in a later version (Clark and Boswell, 1991), thereby enabling it to generate an unordered set of rules. It also replaced the entropy rule evaluation measure with the *Laplace expected error estimate* which is given by:

$$LaplaceAccuracy(n_{class}, n_{covered}, k) = \frac{n_{class} + 1}{n_{covered} + k} \tag{2.10}$$

where

n_{class} = the number of examples of the target class covered by the rule

$n_{covered}$ = the total number of examples covered by the rule

k = the number of classes

The modified algorithm also incorporates a stopping criterion to see if the Laplace estimate of the best complex is better than that of the default rule. If this is the case, the induction continues for the current class. Otherwise, the new complex is not deemed to bring about an improvement and so rule generation for the current class terminates. Since the new version of

CN2 generates an unordered set of rules, so a conflict resolution approach is also adopted in order to resolve any clashes that might occur. In case a new example satisfies more than one rule predicting different classes, a probabilistic method is used in which the distribution of covered examples of each rule among classes is summed to find the most probable class.

FOIL (Quinlan, 1990), which stands for *First Order Inductive Learner,* is a rule induction algorithm that uses a higher representation language which expresses the learned concepts in the form of first-order logic rules. Such a representation is more complex in that the rules may contain variables as opposed to the propositional logic which is used in conventional rule induction algorithms to generate variable-free rules. FOIL uses a greedy search strategy to search through the space of possible concept descriptions. A condition is termed as a *literal* in FOIL whereas the rule evaluation measure used to evaluate a specialised rule formed by addition of a literal is referred to as *FoilGain* and is defined as:

$$FoilGain(literal) = p_{sr}\left(log_2\frac{p_{sr}}{p_{sr}+n_{sr}} - log_2\frac{p_r}{p_r+n_r}\right) \qquad (2.11)$$

where

p_r and n_r = the respective number of positive and negative examples covered by original rule

p_{sr} and n_{sr} = the respective number of positive and negative examples covered by specialised rule

FOIL uses a pre-pruning method based on the minimum description length principle (Rissanen, 1983) in order to prevent overfitting to any noise that may be present in the training data. Using the MDL stopping criterion, the algorithm decides when to stop adding conditions to a rule even if it still covers some negative examples, and when to stop adding rules to the rule set even though all the positive examples have not yet been covered.

RULES (RULe Extraction System) is a family of simple inductive learning algorithms which uses ideas from both *AQ* and *CN2*. The RULES family is different from the other algorithms in that it does not induce rules on a class-per-class basis but considers the class of the selected seed example as the target class. It then attempts to induce a rule that covers as many examples of the target class as possible using the rule evaluation function. As a result, the order of rules appearing in the list follows the order in which uncovered examples are encountered. For instance, the first 5 rules appearing in the rule set produced by the RULES family may belong to class A, C, B, A, and C respectively. Furthermore, the RULES family only marks the examples covered by previous rules instead of removing them.

RULES (Pham and Aksoy, 1993), RULES-2 (Pham and Aksoy, 1995b) and RULES-3 (Pham and Aksoy, 1995a) were the first three algorithms in the family. Later, Pham and Dimov developed a new rule induction algorithm RULES-3 Plus (Pham and Dimov, 1997b) that incorporated the beam search strategy instead of greedy search and used a new rule evaluation measure called H-measure (Lee, 1994) which is defined as:

$$H = \sqrt{\frac{R^c}{N}}\left[2 - 2\sqrt{\frac{R^c_{TC}N_{TC}}{R^cN}} - 2\sqrt{\left(1 - \frac{R^c_{TC}}{R^c}\right)\left(1 - \frac{N_{TC}}{N}\right)}\right] \tag{2.12}$$

where

N = total number of examples in the training data

R^c = total number of examples covered by the rule

N_{TC} = total number of examples of target class in the training data

R_{TC}^{c} = total number of examples of target class covered by the rule

The first term in equation 2.12 represents the generality of the rule whereas the second term in square brackets represents its accuracy.

The next algorithm developed within the RULES family was RULES-4 (Pham and Dimov, 1997a) which follows the incremental learning approach. It makes use of a short-term memory (STM), which is specified by the user and represents the smallest number of training examples required for learning. As new training examples continue to arrive, the population size of the STM increases and as soon as it becomes full, the learning algorithm is activated and starts generating rules.

RULES-5 (Pham et al., 2003) was the next algorithm in the family, the main strength of which is its ability to handle continuous attributes. It also uses H-measure as the rule evaluation measure similar to RULES-3 Plus. However, it employs a more efficient search mechanism as well as a new post-pruning technique (Pham et al., 2004) in order to handle noisy data.

The last algorithm within the RULES family was RULES-6 (Pham and Afify, 2005b) which incorporated several search space pruning techniques in order to cut down computational complexity and to handle noisy data. The rule evaluationmeasure used in RULES-6 is the *m-probability-estimate* (Cestnik, 1990) which is a more general version of the Laplace error estimate and is given by:

$$mAccuracy(n_{class}, n_{covered}, k) = \frac{n_{class} + mP_0(C_t)}{n_{covered} + m} \qquad (2.13)$$

where

$P_0(C_t) = \frac{C_t}{N}$ = probability of target class in the training data

C_t = total number of examples of target class in the training data

N = total number of examples in the training data

m = an estimate of the amount of noise in the data

Figure 2.3 presents a flowchart showing how the RULES family has evolved over the course of time.

2.3.3 Major Contributions on Scalability Issue

Several decision tree algorithms have been proposed in order to address the scalability issue. Some of them assume that the training data can fit in the main memory and focus on the development of data sampling techniques at each node of the tree (Catlett, 1991a) and discretisation techniques (Catlett, 1991b) so as to reduce storage requirements and increase execution speed. Others rely on the use of more efficient data structures in order to handle large datasets that exceed the size of the main memory. There are yet others that use data partitioning methods, feature subset selection techniques or data summarisation techniques in order to cope with large datasets.

SLIQ (Mehta et al., 1996), which stands for *Supervised Learning In Quest,* is a decision tree algorithm that attempts to improve scalability by employing a pre-sorting technique intended to minimise the evaluation cost for continuous attributes. This is coupled with a breadth-first tree growth strategy so as to enable the algorithm to handle disk-resident data. In order to accomplish this, disk-resident *attribute lists* are maintained for all the attributes in the training data along with a memory-resident *class list.* An attribute list stores the values for an attribute

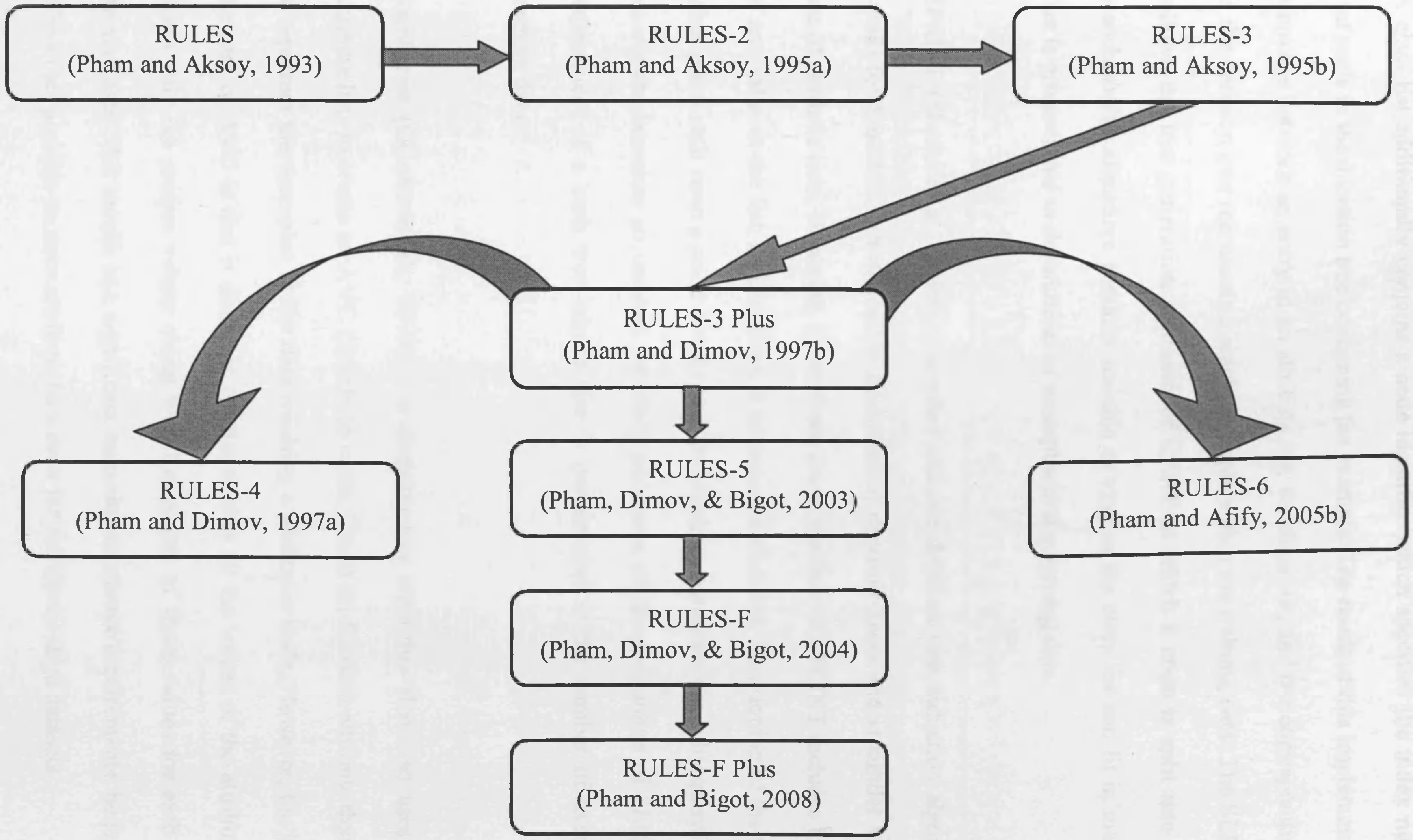


Figure 2.3 Flowchart Describing the Evolution of RULES Family

occurring within the data along with index numbers of the examples containing those values. A class list additionally contains a node identifier which specifies the index number of the leaf node in the decision tree containing the example. The result of this implementation is that a linkage between an entry in an attribute list, the class list, and the corresponding leaf node in the decision tree represents a single example within the training data. The SLIQ algorithm follows the tree generation approach of CART in which a node is split into exactly two branches. The algorithm remains scalable as long as the class list can fit in memory whose size is proportional to the number of examples in the training data.

SPRINT (Shafer et al., 1996) is another scalable decision tree induction algorithm, which stands for *Scalable PaRallelisable INduction of decision Trees* and is similar to SLIQ in its use of attribute lists. However, an attribute list in the case of SPRINT includes the class label of any value in the list, so that there is no separate class list. Partitioning of the attribute lists takes place each time a node is split and the resulting partitions are subsequently distributed among the branches so created. SPRINT addresses all the limitations of SLIQ except the requirement of a hash tree whose size is proportional to the number of examples in the training data.

RainForest (Gehrke et al., 1998) is a decision tree algorithm that also uses the idea of attribute lists to create an AVC (Attribute-value, Class) set for each attribute that can be used to represent the examples in the data reaching a particular node. However, the innovation in the case of AVC is that it does not require storing all the values of the attribute. Instead it stores only the unique values along with the counts of those values for each class in the training data. This results in a significant reduction in storage requirements, because of which it becomes possible to store attribute lists even for extremely large datasets.

PUBLIC (Rastogi and Shim, 1998), which stands for *PrUning and BuiLding Integrated in Classification,* is another scalable decision tree induction algorithm. Instead of pruning the complete tree after it has been grown, PUBLIC calculates a lower bound of the cost of encoding a subtree that would be rooted at a particular node. Based on this estimate, PUBLIC decides whether a node will likely be pruned later on and in that case does not split the node any further. Because of this preventative action, PUBLIC is able to deliver substantial performance improvements over conventional decision tree algorithms.

BOAT (Gehrke et al., 1999) is a decision tree algorithm that handles the issue of scalability in an altogether different manner. Instead of utilising attribute lists or any other data structure for that matter, it uses the "bootstrapping" statistical technique which requires drawing out a number of small subsets from the training data, each of which can fit in the main memory. The algorithm is named after this statistical technique and stands for *Bootstrapped Optimistic Algorithm for Tree Construction.* Each of the subsets is passed to the learning algorithm which then constructs a decision tree from it, and the resulting trees are combined to generate the final tree. The algorithm continues to refine this tree as long as it does not closely match the tree that would have been generated from the whole training data based on a lower bound on the splitting criterion. BOAT has been found to be a significant improvement over RainForest and other early decision tree induction algorithms in that it constructs several levels of the tree in only two scans of the training data. Furthermore, it also supports incremental learning so that the existing tree can be updated in case the dataset changes dynamically over time.

Recently, a new decision tree algorithm SURPASS (Li, 2005), which stands for *Scaling Up Recursive Partitioning with Sufficient Statistics,* has been proposed. The algorithm incorporates linear discriminant analysis (LDA) into the recursive partitioning process of generating a decision tree. It summarises all the information required for tree generation into a set of sufficient statistics, which are expressed in the form of a few vectors and matrices, and whose size is determined by the number of attributes and classes in the dataset. The set of sufficient statistics can be gathered incrementally from the data, by reading a subset of the data from the disk to main memory one at a time. Consequently, the size of the dataset that can be handled by this algorithm is independent of memory size. SURPASS has been shown to produce decision trees that not only require a lower execution time as compared to the RainForest algorithm but are also competitive with respect to error rate.

Contrary to decision tree learning, very few of the rule induction algorithms proposed to date have used any special data structures for the sake of improving scalability. Instead, most scalable rule induction algorithms have focused on the use of pruning techniques in order to enable them to handle large datasets. One of the first rule induction algorithms that could scale up to large datasets was IREP (Furnkranz and Widmer, 1994), which stands for *Incremental Reduced Error Pruning*. The algorithm improves upon a previous post-pruning rule induction algorithm REP (Brunk and Pazzani, 1991) which prunes the complete rule set after it has been generated. IREP however is incremental in the sense that it prunes a single rule immediately after it has been generated. This leads to the removal of a greater number of examples covered by the rule so generalised, which speeds up induction as fewer and fewer examples remain to be covered.

Despite the fact that IREP is scalable, its classification accuracy has been found to be inferior to that of C4.5 for a variety of domains. In order to address this issue, a new algorithm RIPPER (Cohen, 1995), which stands for *Repeated Incremental Pruning to Produce Error*

Reduction, was proposed. The algorithm uses a new rule evaluation measure and generates rules on a class-per-class basis starting with the class with the lowest frequency. Rules for a class are generated using the IREP algorithm and a stopping criterion based on the MDL principle decides when to stop inducing rules for the current class. This is followed by an optimisation phase which uses the full dataset to first grow a replacement rule and then a revised rule for each rule in the rule set. The evaluation is carried out using the usual reduced error pruning, with the exception that the generated variations are evaluated on a subset of the pruning set which does not include the examples covered by the other rules. The algorithm then considers all three rules including the original rule, the replacement rule, and the revised rule, and keeps the one with the smallest description length in the context of the rule set. It then induces residual rules in order to cover examples that have not already been covered by the rule set formed so far. Finally, it checks whether each rule contributes to the overall reduction of the description length before generating rules for the next class. RIPPER has been shown to be as efficient as IREP and at the same time competitive with C4.5 in terms of classification accuracy.

More recently, addressing the issue of scalability for rule induction algorithms has been attempted in RULES-6 (Pham and Afify, 2005b). It uses the concept of invalid attribute values in order to weed out parts of the search space that are deemed not to be solutions. In order to accomplish this, RULES-6 requires three different inputs from the user, namely the minimum positives, minimum negatives, and consistency of the specialised rule. In case any of these criteria are not satisfied by the specialised rule, the last attribute value conjunction used to form it is added to the invalid values set of its parent. This ensures that the same attribute value is not used to specialise other descendents of that parent, which improves the speed of execution.

2.4 Essentials of Association Learning

Association rule mining (ARM) originated in the context of market basket analysis with the intent to discover sets of items that are frequently purchased by shoppers. The associations derived from such frequent itemsets occurring in massive transactional records can help many businesses in the decision-making process which may include catalog design, cross-marketing, and customer shopping behavior analysis (Han and Kamber, 2006). It is similar to classification learning in that the end goal in both is to mine a set of rules. However, association rule mining falls under the category of unsupervised learning since there is no class in a transactional database as opposed to the data used for classification learning. Another difference is that in the case of classification learning the rules for a particular class unanimously try to achieve the objective of separating the examples of that class from those of other classes in the dataset. This is not true in the case of association learning in which each rule is a stand-alone piece of information that represents a particular regularity or pattern underlying the transactional database.

2.4.1 Frequent Itemset Mining

The major bottleneck in the case of association learning is the *frequent itemset mining* (FIM) phase, which discovers frequent itemsets in a transactional database, following which the generation of association rules is pretty straightforward. The frequent pattern analysis does not take into account the quantity of a purchased item, which means that it considers only the unique items present in a transaction. An *itemset* is simply a collection of one or more items, so a *k-itemset* implies that the itemset has *k* items in it. A *frequent itemset* is defined as one that occurs in a certain minimum number of transactions in the database. This is dictated by the user-specified *min-support* which is expressed as a percentage of the total number of transactions in the database. Since the number of candidate frequent itemsets is exponential to the total number of items present in the transactional database, an efficient frequent itemset mining algorithm is one that examines as few candidate frequent itemsets as possible (Kantardzic, 2003).

2.4.1.1 Early Frequent Itemset Mining

One of the first ARM algorithms was Apriori (Agrawal et al., 1993, Agrawal and Srikant, 1994) which uses a level-wise search to find the set of frequent itemsets occurring in the database. The length of the frequent itemsets discovered by Apriori increases by one in each scan of the database, i.e. it finds 1-itemsets in the first scan, 2-itemsets in the second scan and so on. It uses the anti-monotonic property in subsequent scans in order to weed out a candidate itemset if any of its subsets turns out to be infrequent, which means that the subset is not present in the set of frequent itemsets discovered in the previous scan. This increases the speed of execution of the algorithm significantly.

The inefficiency of the Apriori algorithm at mining extremely large transactional databases owes much to the fact that it also counts the support of those infrequent candidate itemsets that cannot be eliminated on the basis of subset pruning (Kantardzic, 2003). In order to handle this issue, newer frequent itemset mining methods have been sought that can eliminate as many infrequent candidate itemsets from consideration as possible. A few attempts in this direction include the Partition algorithm (Savasere et al., 1995), the DIC algorithm for *Dynamic Itemset Counting* (Brin et al., 1997), and the ECLAT algorithm (Zaki et al., 1997), which stands for *Equivalence CLASS Transformation.*

The Partition algorithm subdivides the transactional database D into n nonoverlapping partitions. It then scans the database to find the frequent itemsets local to each partition. Since the discovered itemsets are only candidate frequent itemsets globally, the database is scanned for a second time to determine the actual support of each in order to find the global frequent itemsets. The DIC algorithm, on the contrary, is eager and can discover new candidate itemsets dynamically as the database scan progresses for calculating the support of previously found candidate itemsets. It does that by estimating the support of all the itemsets that have been counted so far, and adding new candidate itemsets as soon as all of their subsets are found to be frequent. The ECLAT algorithm uses a vertical database layout and requires only a single scan of the database. It uses novel itemset clustering techniques based on equivalence classes and maximal hypergraph cliques combined with efficient lattice traversal techniques to generate the frequent itemsets contained in each cluster.

2.4.1.2 Closed and Maximal Frequent Itemset Mining

Despite the subset pruning technique used in the above mentioned association learning algorithms, the search space in case of gigantic transactional databases is still challenging enough to call for the development of new algorithms that can mine the complete set of frequent itemsets without incurring high computational cost. Consequently, alternative frequent itemset mining methods emerged that mine only a subset of the set of all frequent itemsets, namely the *closed frequent itemsets* and the *maximal frequent itemsets* as shown in Figure 2.4.

To illustrate the concept of closed and maximal frequent itemsets, Table 2.4 shows an example transactional database with a total of five items in it. The itemset lattice for the database is shown in Figure 2.5. The numbers appearing above the ovals in the figure represent the IDs of the transactions in which the itemset appears. The min-support turns out to be 2 as per the support percentage specified by the user and so any itemset that appears in

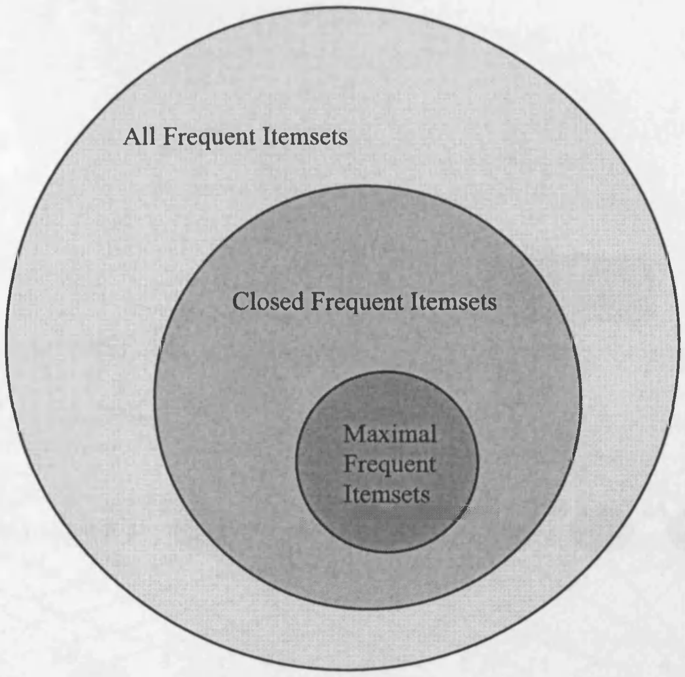


Figure 2.4 The Concept of Closed and Maximal Frequent Itemsets.

TID	Items Present in the Transaction
1	A B C
2	A B C D
3	B C E
4	A C D E
5	D E

Table 2.4 An Example Transactional Database with Five Items.

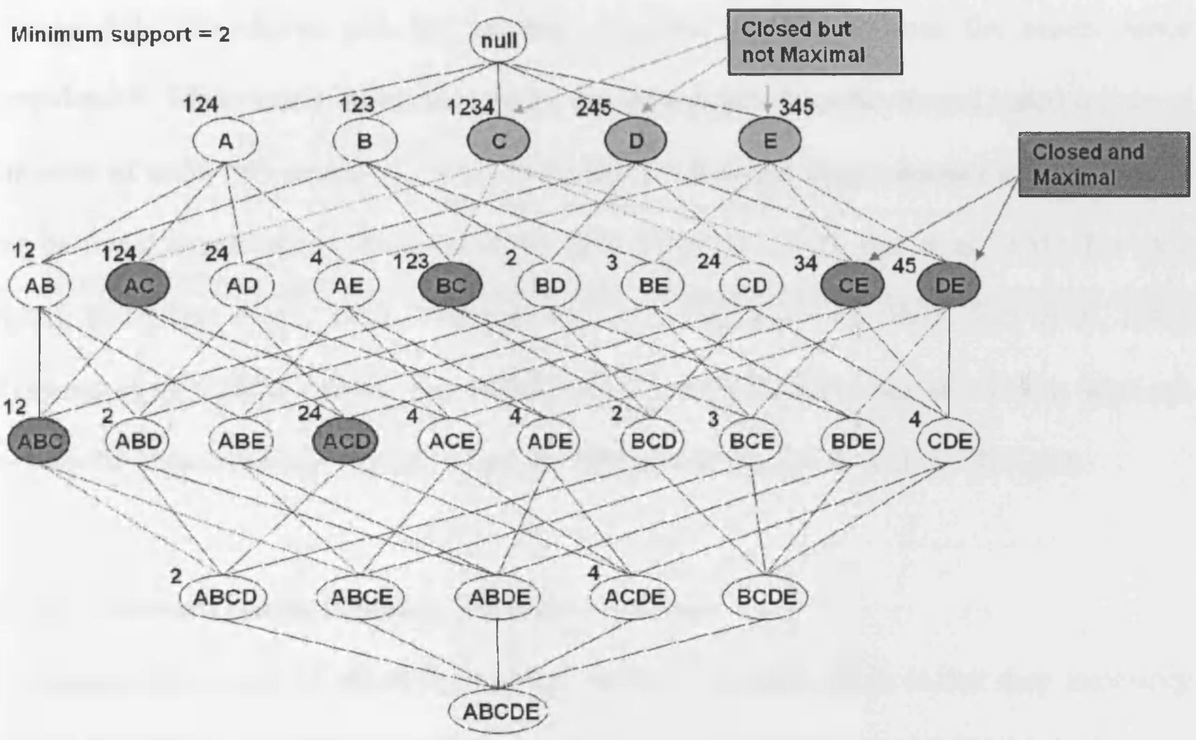


Figure 2.5 The Itemset Lattice Illustrating Closed and Maximal Frequent Itemsets.

two or more transactions is a frequent itemset. The closed and the maximal frequent itemsets have been highlighted in the figure, with the former appearing in level I only and the latter appearing in both level II and level III. It is clear from the lattice that an itemset is closed if the support of all its immediate supersets is less than itemset support, and additionally it is maximal if the support of all its immediate supersets is less than min-support.

Mining only the closed and the maximal frequent itemsets reduces the search space considerably. Many association rule mining algorithms have been developed based on one or the other of these two strategies. Those that adopt the frequent closed itemset mining strategy can be found in references (Pasquier et al., 1999, Pei et al., 2000, Han et al., 2002, Zaki and Hsiao, 2002, Pan et al., 2003, Wang et al., 2003, Yan and Han, 2003, Yan et al., 2003, Jianyong et al., 2005). Others that exploit the maximal frequent itemset mining approach include the Max-Miner algorithm (Bayardo, 1998), MAFIA (Burdick et al., 2001) etc.

2.4.1.3 Frequent Itemset Mining Without Candidates

An inherent limitation of all of the mining methods outlined above is that they inevitably require generating candidate frequent itemsets and counting their support before discovering the actual frequent itemsets. A novel frequent itemset mining algorithm that can mine the complete set of frequent itemsets without candidate generation is known as *FP-growth* (Han et al., 2000), which stands for *frequent pattern growth.* It adopts a *divide-and-conquer* strategy to compress the transactional database into an equivalent FP-tree which is then traversed to mine the frequent itemsets. This is accomplished by scanning the database once in order to find the frequent 1-itemsets L and sorting them in support descending order. Following this, the database is scanned a second time, in which one branch of the tree is created for each transaction by reading the items in it in the same order as that of L. This results in a tree in which items appear in a most frequent to least frequent order from top to the bottom. In order to link the same items appearing in the tree, an item header table is maintained in which each item is linked to its occurrences in the tree by means of a chain of node-links. To find the set of frequent itemsets, the FP-tree is mined in a bottom-up direction starting with each 1-itemset as an initial *suffix* and constructing its conditional pattern base, which consists of the set of *prefix paths* in the FP-tree occurring with the suffix pattern. This is followed by constructing a *conditional* FP-tree of the suffix and mining recursively on the tree. Finally, pattern growth is achieved by concatenation of the suffix pattern with the frequent patterns generated from the conditional FP-tree. The FP-growth algorithm has been shown experimentally to be faster than the *Apriori* algorithm by about an order of magnitude.

2.4.2 Generation of Association Rules

The generation of association rules is pretty straightforward once the frequent itemsets have been discovered. In order to generate one or more association rules, a frequent itemset must have some nonempty subsets. This means that only frequent i-itemsets with $i = 2, 3, \ldots$ can be used to generate association rules. To generate association rules from any frequent i-itemset, the following rule is adopted:

Subset-of-Itemset → Itemset − Subset-of-Itemset

For example, the nonempty subsets of frequent 3-itemset {I1, I2, I3} are:
{I1}, {I2}, {I3}, {I1, I2}, {I1, I3}, and {I2, I3}

Hence some of the association rules that can be formed from this itemset are:

$I1 \rightarrow I2 \land I3$

$I2 \land I3 \rightarrow I1$

and so on …

The *support* of an association rule corresponds to the coverage of a rule in classification learning, whereas *confidence* corresponds to accuracy or consistency. Both of these are regarded as statistical measures of rule interestingness. They have been used successfully by all association rule mining algorithms that generate candidate frequent itemsets to minimise combinatorial explosion by restricting the search to interesting patterns only.

2.5 New Approaches to Classification Learning

Recently, a newer approach to generating classification rules has been developed, referred to as *associative classification.* It mines frequent itemsets from the database in the first phase using support and confidence as interestingness measures of a pattern. An item corresponds to an attribute-value pair in the dataset on which classification is to be performed. In the second phase however, the focus is kept on mining association rules with only a single item in their consequent, i.e. rules of the form $I1 \land I2 \land \ldots \longrightarrow A_c = C$, where A_c is the classification attribute and C is any class label belonging to it. This is in contrast to association rule generation in which a rule may have any number of items both in its antecedent and consequent.

The new method of mining classification rules was initially proposed in the CBA algorithm (Liu et al., 1998), which stands for *Classification Based on Association,* and presents a framework for integrating classification and association rule mining. It uses an approach similar to the Apriori algorithm in the first phase, using a minimum support threshold for mining the frequent itemsets (referred to as *ruleitems* in CBA because of a single consequent item). From the mined ruleitems, those that also satisfy a minimum confidence threshold are selected as the *class association rules* (CARs). In the second phase, the generated CARs are analyzed to build the final classifier. The algorithm sorts the generated rules by their support and confidence, so the rule set generated by CBA is a decision list. Furthermore, it uses a heuristic method for building the classifier. CBA was shown to be an improvement over C4.5 based on its empirical evaluation on 25 datasets.

Later, a new algorithm CMAR (Li et al., 2001), for *Classification based on Multiple Association Rules,* was proposed. Instead of mining association rules directly from the database, CMAR adopts an enhanced version of the FP-growth algorithm which creates an FP-tree to register not only all of the frequent itemset information but also the distribution of class labels among examples satisfying any frequent itemsets. The result is the integration of the frequent itemset mining and rule generation phases. Along with the use of confidence and support as rule interestingness measures, CMAR also uses a χ^2 test of statistical significance as a pruning technique to see if the antecedent of the rule and its class are positively correlated. When predicting the class of a new example, CMAR bases its decision on multiple association rules, as opposed to CBA which assigns it the class label of the most confident rule among the rule set. This is accomplished by grouping together the rules belonging to a class and using a weighted χ^2 measure to find the strongest group of rules within a group whose class label is then assigned to the example. CMAR was found to be a little more accurate than CBA as well as much more scalable and efficient.

In order to avoid generating a large number of rules as in associative classification, another algorithm CPAR (Yin and Han, 2003) was proposed later which attempts to combine the good features of both associative classification and traditional rule induction algorithms.

CPAR stands for *Classification Based on Predictive Association Rules*. It generates rules by following an approach similar to the FOIL algorithm and integrates the features of associative classification in predictive rule analysis. However, instead of removing the examples covered by a rule, CPAR allows the covered examples to remain under consideration by only reducing their weight. The algorithm also generates and tests more rules than traditional rule-based classifiers in order to avoid missing important ones. To avoid overfitting, it uses the Laplace expected error estimate to evaluate each rule and uses the best *k* rules in prediction instead of all of the rules of a group so as to avoid the influence of lower ranked rules.

HARMONY (Wang and Karypis, 2006), which stands for *Highest confidence clAssification Rule Mining fOr iNstance-centric classifying,* is another algorithm that attempts to overcome the problems of both rule-induction-based and association-rule-based algorithms. For each example in the training dataset, the algorithm directly mines one of the *Highest Confidence Classification Rules HCCR* that it supports along with satisfying a user-specified minimum support constraint. The classification model is built by uniting these rules over the entire set of instances. HARMONY employs an instance-centric rule generation framework which ensures the inclusion of the best possible rule for each example. Since an example in the training data can satisfy many of the discovered rules, the resulting classifier has the potential to achieve better classification accuracy because of its ability to generalise to new examples. Furthermore, because of using several novel search strategies and pruning techniques, the algorithm is also efficient and scalable.

2.6 Summary

This chapter has outlined basic classification learning concepts including representation of the input data as well as the two main approaches for expressing the learned classifier in the context of inductive learning. It has then discussed the different schemes for evaluating the classification accuracy of the learned classifier. Following this, representative algorithms of the two major inductive learning paradigms have been described in detail along with a discussion of algorithms developed for handling the scalability issue. The chapter has also focused on association learning, a data mining activity closely related to classification learning. Finally, a new area, referred to as associative classification, has been discussed as well as some other recent approaches that attempt to integrate the good features of both association and classification learning.

CHAPTER 3

RULES-7: AN EFFICIENT NOISE TOLERANT RULE INDUCTION ALGORITHM

3.1 Motivation

Real-world applications of data mining involve huge datasets that may have billions of training examples and thousands of attributes. Several representative examples of such datasets have been cited in (Fayyad et al., 1996) where the size of the data is often in gigabytes and sometimes even in terabytes. Besides, the datasets for classification learning may contain hundreds or thousands of classes. To make matters worse, the presence of noise in such gigantic datasets makes the discovery of accurate and reliable patterns an even more challenging task for the learning algorithm. Mining such huge noisy repositories of information therefore necessitates the development of scalable and robust rule induction algorithms with a runtime that is not only predictable but also acceptable to the end-user.

Of the many approaches that may be taken to improve the scalability of rule induction algorithms, the most straightforward one is the development of efficient algorithms that can process huge amounts of data in a relatively short time. One way to achieve this may be the use of an efficient search strategy such as greedy search which avoids complexity by looking only for a single good solution during its search for the best rule. However, because of its inability to recover from any ill-considered judgments made along the way, it may not be the optimal choice for algorithms that seek to mine a reliable classification model. Alternatively, beam search, which retains a set of active substitute solutions, may be used along with some pruning techniques designed to discard portions of the search space. This is exactly the strategy used in the RULES-6 algorithm (Pham and Afify, 2005b), which was designed to handle large noisy datasets effectively using certain user-specified *search space pruning* techniques. The algorithm was built on its predecessor RULES-3 Plus (Pham and Dimov, 1997b) which fails to cope with large and noisy datasets because of not employing any pruning techniques and instead emphasising completeness and consistency.

Association rule mining algorithms (Agrawal et al., 1993, Agrawal and Srikant, 1994) on the other hand have been inherently capable of dealing with large datasets ever since their emergence (Witten and Frank, 2005). The Apriori algorithm, for example, initially used the support of a pattern as an interestingness measure in order to cut down on the potentially exponential number of patterns. It then used the confidence measure in order to weed out the discovered association rules. The goal of such interestingness measures in the context of association rule learning is to restrict the search space to only rules that are of interest to the end-user. Association learning however seeks to find only the frequent (interesting) patterns that can then be used to express strong associations between items in a transactional database. Classification learning by contrast requires a discriminant function in order to identify a rule that can separate examples of a class from those of other classes. Consequently, the mere coverage or accuracy of a rule is not enough in classification learning in order to discover rules that are maximally general as well as accurate. However, the support-based interestingness measure from association learning can be used in classification as long as the discriminant function is subordinate to this constraint.

This chapter proposes a new algorithm that improves the search strategy developed in the RULES-6 algorithm. The most significant improvement is applied to the control structure for implementing the search space pruning techniques proposed in RULES-6. Because of this, the algorithm does not fully achieve the real objective of pruning any over-specific rules in order to avoid overfitting along with cutting down the search space. Instead, changing the user-specified parameters employed in the search space pruning steps results only in a minute reduction in the learning time. Secondly, because the duplicate rule handling strategy was not implemented in the most efficient way, the reduction in the learning time of RULES-6 is not as much as it can be. Thirdly, there is an assumption regarding the value of the noise parameter in the *specialisation measure* employed in RULES-6, which can lead to over-general rule sets even in case of datasets with little noise.

This chapter is organised as follows. Section 3.2 gives a brief description of RULES-6, the most recent algorithm in the RULES family of inductive learning algorithms. Section 3.3 discusses in detail the problems identified with RULES-6 along with the modifications suggested for inclusion in RULES-7. Section 3.4 presents a detailed description of RULES-7 and discusses a new comparison heuristic incorporated into the algorithm as well as two new pruning techniques based on ideas from association learning. Section 3.5 presents the results obtained from experimental evaluation of RULES-7 on some benchmark datasets. Section 3.6 summarises and concludes the chapter.

3.2 The RULES-6 Algorithm

RULES-6 (RULe Extraction System – Version 6) like its predecessor RULES-3 Plus, performs a general-to-specific beam search in order to induce a rule from a seed instance. The selection of the seed is random, meaning that seeds are selected sequentially starting with the first instance in the training dataset. The default values of *w*, MinNegatives, and MinPositives for RULES-6 are 4, 1, and 2 respectively (Pham and Afify, 2005b). A pseudo-code description of RULES-6 is given in Figure 3.1 and the *Induce_One_Rule* procedure is outlined in Figure 3.2. It should be noted that the original terminology used in RULES-6 was modified for clarity and is shown in Table 3.1.

The selection of the seed is done at step 4 of the *Induce_Rules* procedure, following which the control is transferred to the *Induce_One_Rule* procedure. After initialising the *ParentRuleSet* and *ChildRuleSet* to empty, the *Induce_One_Rule* procedure removes all conditions from the antecedent of the rule, leaving only the class of the selected seed example in its consequent. This most general rule is then declared as the *BestRule* and is added to the *ParentRuleSet* for specialisation. The While loop at step 2 is subsequently activated and continues to run until the *ParentRuleSet* becomes empty again.

The *BestRule* is now specialised at step 3 by adding conditions to its antecedent each of which corresponds to an attribute-value in the selected seed example. This results in *n* *ChildRules* initially, each with a single condition in its antecedent, where *n* is the number of attribute-values in the seed example. At step 4, as opposed to its predecessor RULES-3 Plus which prefers consistency over generality, the RULES-6 algorithm declares the *ChildRule* as the *BestRule* only if its specialisation measure is greater than the last *BestRule.*

If any *ChildRule* satisfies any one of the three search space pruning steps 5, 6 and 7, it is pruned and the last value used to specialise it is added to the InvalidValues set of its *ParentRule*. However, this *ChildRule* might already have replaced the last *BestRule,* as in the case that its score was greater, even if it now satisfies any of the search space pruning steps.

Step 10 further attempts to prune rules from the *ChildRuleSet* based on whether the

```
Procedure Induce_Rules (TrainingSet, BeamWidth)
RuleSet = ∅                                                          (step 1)
While all the examples in the TrainingSet are not covered Do         (step 2)
  Take a seed example s that has not yet been covered.               (step 3)
  Rule = Induce_One_Rule (s, TrainingSet, BeamWidth)                 (step 4)
  Mark the examples covered by Rule as covered.                      (step 5)
  RuleSet = RuleSet ∪ {Rule}                                         (step 6)
End While
Return RuleSet                                                       (step 7)
End                                                                  (step 8)
```

Figure 3.1 A pseudo-code description of RULES-6

```
The Induce_One_Rule Procedure
Procedure Induce_One_Rule (s: Seed example, Instances: Training set, w: Beam width)
ParentRuleSet = ChildRuleSet = $\emptyset$
BestRule = most general rule (the rule with no conditions)                      (step 1)
ParentRuleSet = ParentRuleSet $\cup$ {BestRule}
While ParentRuleSet $\neq \emptyset$ Do                                          (step 2)
  For each ParentRule $\in$ ParentRuleSet Do
    For each nominal attribute $A_i$ that does not appear in ParentRule Do
      If $v_{is} \in$ ParentRule.ValidValues, where $v_{is}$ is the value of $A_i$ in s Then
        ChildRule = ParentRule $\wedge$ [$A_i = v_{is}$]                         (step 3)
        If ChildRule.Score > BestRule.Score Then                                 (step 4)
          BestRule = ChildRule
        If ChildRule.Classified < MinPositives OR                                (step 5)
          ParentRule.Misclassified – ChildRule.Misclassified < MinNegatives OR
                                                                                 (step 6)
          ChildRule.Consistency = 100% Then                                      (step 7)
          ParentRule.InvalidValues = ParentRule.InvalidValues + {$v_{is}$}
                                                                                 (step 8)
        Else
          ChildRuleSet = ChildRuleSet $\cup$ {ChildRule}                         (step 9)
    End For
  End For
  Empty ParentRuleSet
  For each ChildRule $\in$ ChildRuleSet Do
    If ChildRule.OptimisticScore $\leq$ BestRule.Score Then                      (step 10)
      ChildRuleSet = ChildRuleSet – {ChildRule}                                  (step 11)
      ParentRule.InvalidValues = ParentRule.InvalidValues + Last_Value_Added
                                                              (ChildRule)        (step 12)
End For
```

Figure 3.2 A pseudo-code description of the *Induce_One_Rule* procedure of RULES-6

```
    For each ChildRule ∈ ChildRuleSet Do
      ChildRule.ValidValues = ChildRule.ValidValues − ParentRule.InvalidValues
                                                                          (step 13)
    End For

    If $w > 1$ Then
      Remove from ChildRuleSet all duplicate rules
      Select $w$ best rules from ChildRuleSet and insert into ParentRuleSet    (step 14)
      Remove all rules from ChildRuleSet
  End While
  Return BestRule
```

Figure 3.2 A pseudo-code description of the *Induce_One_Rule* procedure of RULES-6 (continued).

No.	Old Terminology	New Terminology
1.	*PartialRules:*	*ParentRuleSet*
2.	*NewPartialRules*	*ChildRuleSet*
Reason.	The terms *PartialRules* and *NewPartialRules* do not indicate that they are RuleSets and also give no clue as to the *Parent-Child* relationship between these two RuleSets.	
3.	*Rule* and *NewRule:*	*ChildRule*
Reason.	The terms *Rule* (in *Rule* ∈ *NewPartialRules*) and *NewRule* do not indicate that the rule is a *ChildRule*, i.e. a specialisation of the *ParentRule*.	
4.	*Rule* and *Parent (NewRule):*	*ParentRule*
Reason.	The terms *Rule* (in *Rule* ∈ *PartialRules*) and Parent (*NewRule*) both stand for the *ParentRule*. So using two different terms is obviously confusing.	
5.	*MinNegatives:*	*MinExcludedNeg*
Reason.	One might think that MinNegatives indicates the minimum number of Negative instances that the *ChildRule* should ***cover***. However, this is not the case as MinNegatives is the minimum number of Negative instances that the *ChildRule* should ***exclude*** with respect to its *ParentRule* in order to qualify for addition to the *ChildRuleSet* for further specialisation.	
6.	*Covered_Positives:*	*Classified*
7.	*Covered_Negatives:*	*Misclassified*
Reason.	The instances which are *Covered* by the *ChildRule* and are *Positive* (i.e. belong to the target class of *ChildRule*) are obviously those which are *Classified* by the *ChildRule*. Likewise, the instances which are *Covered* by the *ChildRule* but are *Negative* (i.e. do not belong to the target class of *ChildRule*) are obviously those which are *Misclassified* by the *ChildRule*.	

Table 3.1 The Modified Terminology

optimistic score of the rule is less than or equal to the score of the *BestRule*, in which case the last value used for specialisation is added to the InvalidValues set of its *ParentRule*. Step 13 removes from the ValidValues set of each *ChildRule* the InvalidValues set of its *ParentRule*. Finally, based on whether the beam width w is greater than 1, step 14 removes from the *ChildRuleSet* all duplicate rules, copies w *BestRules* from the *ChildRuleSet* into the *ParentRuleSet*, empties the *ChildRuleSet* and transfers the control to the While loop at step 2 to repeat the specialisation iteration. The rules in the *ParentRuleSet* with n condition(s) in their antecedent now act as *ParentRules* to produce *ChildRules* with $n + 1$ conditions in their antecedent.

When there are no rules left in the *ChildRuleSet* to be copied into the *ParentRuleSet*, the While loop at step 2 terminates and the *BestRule* is returned to the *Induce_Rules* procedure. The instances covered by the *Rule* are marked at step 5, the *Rule* is added to the *RuleSet*, and the control is transferred to the While loop at step 2 in order to select another seed example and pass it to the *Induce_One_Rule* procedure. The final *RuleSet* is returned when no uncovered instances are left in the training dataset.

3.3 Problems Identified with RULES-6

The problems pinpointed with the RULES-6 algorithm will be outlined in the same order as that followed by the algorithm description. For a vivid illustration of these problems, the *weather dataset* shown in Table 3.2 is used since it is small enough to capture the essence of the problems. The attribute "A_5 = Play" is taken as the classification attribute.

Attributes → Examples ↓	**A_1 = Outlook**	**A_2 = Temperature**	**A_3 = Humidity**	**A_4 = Windy**	**A_5 = Play**
1.	sunny	hot	high	false	**no**
2.	sunny	hot	high	true	**no**
3.	overcast	hot	high	false	**yes**
4.	rainy	mild	high	false	**yes**
5.	rainy	cool	normal	false	**yes**
6.	rainy	cool	normal	true	**no**
7.	overcast	cool	normal	true	**yes**
8.	sunny	mild	high	false	**no**
9.	sunny	cool	normal	false	**yes**
10.	rainy	mild	normal	false	**yes**
11.	sunny	mild	normal	true	**yes**
12.	overcast	mild	high	true	**yes**
13.	overcast	hot	normal	false	**yes**
14.	rainy	mild	high	true	**no**

Table 3.2 . The Weather Dataset [Witten and Frank, 2005].

3.3.1 Duplicate Candidate Rules

The first problem identified with the RULES-6 algorithm is that of testing duplicate candidate rules. An example of a duplicate rule is:

IF *Outlook* = *sunny* **AND** *Humidity* = *normal* **THEN** *Class* = *yes* R1

IF *Humidity* = *normal* **AND** *Outlook* = *sunny* **THEN** *Class* = *yes* R1.duplicate

The approach which should be adopted for handling duplicate rules is to scan the *ChildRuleSet* set to check if the generated *NewRule* is already present in the list. If so, then the processing of the candidate rule should be skipped because that rule has already been processed.

RULES-6 does not perform this check when a new candidate rule is generated. As a result, the entire processing that was carried out for R1 is repeated for R1.duplicate. Finally, after spending the time for processing each rule Ri and its duplicates, RULES-6 removes the duplicate rules from the *ChildRuleSet* in step 14 as mentioned earlier.

However, it is obvious that duplicate rule removal is more complicated than duplicate rule avoidance. This is because the latter only requires scanning of the *ChildRuleSet.* In contrast, apart from scanning the *ChildRuleSet,* step 14 also requires a replacement step in which the duplicate rule is overwritten with the *HeadRule* (the first rule in *ChildRuleSet*) and a deletion step in which the *HeadRule* is removed.

In order to avoid the above-mentioned issue, RULES-7 employs the duplicate rule avoidance step after step 3 to prevent duplicate rules from being added to the *ChildRuleSet.* An obvious advantage of this is the reduction in execution time, particularly when the *beam width w* is higher than 1. At a beam width $w = 1$, only single condition rules (highly general rules) are induced such as:

IF *Outlook = sunny* **THEN** *Class = yes* R2

IF *Humidity = normal* **THEN** *Class = yes* R3

...

in which case there is no possibility of any duplication. The problem occurs only when there are more than 2 conditions in the antecedent of the rule, i.e. when $w \geq 2$. A higher value of beam width often results in increased training accuracy for data sets with many attributes, by giving the learning system *w* more chances to find the *BestRule*. Tests conducted in section 3.5.1 on a total of 7 datasets using beam width values of $w = \{4, 6, 8, 10, 12\}$ indicate that the approach can reduce the learning time by as much as 40%. Furthermore, the increase in speed is even higher at larger beam widths.

3.3.2 Enhancement of the Control Structure

The second improvement of the RULES-6 algorithm takes place in the control structure shown in Figure 3.3, which includes the *rule comparison* step 4 and the *search space pruning* steps 5, 6 and 7. The motive behind using the search space pruning steps is to check if any *ChildRule* of a *ParentRule* should not be further specialised. It is the case if a *ChildRule* does not satisfy the user-specified MinPositives constraint or MinExcludedNeg constraint with respect to its *ParentRule*, or if it becomes consistent thereby requiring no further specialisation.

```
For each nominal attribute $A_i$ that does not appear in ParentRule Do
If $v_{is} \in$ ParentRule.ValidValues, where $v_{is}$ is the value of $A_i$ in $s$ Then
   ChildRule = ParentRule $\wedge$ [$A_i = v_{is}$]                                  (step 3)
If ChildRule.Score > BestRule.Score Then                                             (step 4)
   BestRule = ChildRule
If ChildRule.Classified < MinPositives OR                                            (step 5)
  ParentRule.Misclassified − ChildRule.Misclassified < MinExcludedNeg OR             (step 6)
  ChildRule.Consistency = 100% Then                                                  (step 7)
  ParentRule.InvalidValues = ParentRule.InvalidValues + {$v_{is}$}                   (step 8)
Else
  ChildRuleSet = ChildRuleSet $\cup$ {ChildRule}                                     (step 9)
```

Figure 3.3 RULES-6 Search Space Pruning Control Structure

The control structure in Figure 3.3 performs the rule comparison step before the search space pruning, thus allowing any created *ChildeRule* to be considered as *BestRule* regardless of the user specified criteria MinPositives and MinExcludedNeg. The assumption behind this is that the prime criterion for rule selection should be the score. This might result in the selection of rules with high score that are also very specific. Such rules are therefore considered relevant because no other created general rules could beat their score.

However, it could be argued that while increasing the rule set size, these created specific rules might not perform well when facing unseen examples. This is because the rule comparison step is outside the search space pruning steps, so the user-specified MinPositives and MinExcludedNeg constraints fall short of achieving their primary objective, *viz to prune any over-specific rules in order to avoid overfitting* together with cutting down the search space. In this research, it is considered that the foremost criteria to be satisfied by the *ChildRule* should be the user-specified MinPositives and MinExcludedNeg constraints, and only if the *ChildRule* satisfies these constraints should its score be calculated and compared with the last *BestRule*. As a result, it can be argued that RULES-6, by comparing the score of every *ChildRule* with the last *BestRule*, does not take full advantage of the user-specified constraints. Instead, the only noticeable difference arising from allowing MinPositives and MinNegatives to vary is a slight increase in the speed of the algorithm.

In addition, as mentioned previously, the final rule set includes *many specific rules* which do not satisfy the user-specified MinPositives criterion, which is an indicator of the amount of noise in the data. For example, if MinPositives = 5 then the final rule set includes quite a number of rules that cover less than 5 positive instances as well as many others which cover only 1-2 instances. Since the main purpose of the user-specified MinPositives criterion is to filter any rules that cover less than the minimum positive instances in order to speed up the search for genuine reliable rules, such rules are clearly not acceptable to the user as they are most likely based on noise. Hence, such rules do not reflect true regularities contained within the dataset and will certainly not perform well on unseen data.

3.3.2.1 Rectification

In the light of the above discussion, it is necessary that the control structure should be modified so that rule comparison is performed after constraint checking. However, consistency checking (step 7) should still take place after rule comparison because a *ChildRule* could still be labelled the new *BestRule* regardless of whether or not it is consistent. This is to avoid favouring consistency over generality and to improve noise tolerance. The modified control structure for RULES-7 is shown in Figure 3.4.

The result of this change is that only a *ChildRule* satisfying the MinPositives and MinNegatives constraints and scoring higher than the last *BestRule* will be permitted to replace it. Otherwise, the rule will not be added to the *ChildRuleSet* and the last attribute-value pair used to form the *ChildRule* will be added to the InvalidValues set of its *ParentRule*. This is to prevent the algorithm from trying to include that attribute-value pair in the next iteration in the remaining 'siblings' of *ChildRule* (those added to the *ChildRuleSet*). If the *ChildRule* successfully crosses the filter and is adopted as the new *BestRule*, its consistency will be checked. If the consistency is found equal to 100%, the InvalidValues step will be executed again to ensure that the attribute-value pair will not be used to specialise the remaining *ChildRules* in the next iteration. The reason is that appending such an attribute-value pair to the remaining *ChildRules* will only result in consistent rules which are as

```
For each nominal $\text{Condition}_i$ that does not appear in ParentRule
If $\text{Condition}_i$ is not marked as "Invalid" for ParentRule Then
  ChildRule = ParentRule ∧ [$\text{Condition}_i$]                                   (step 3)
 If ChildRule.Classified ≥ MinPositives AND                                        (step 4)
   ParentRule.Misclassified − ChildRule.Misclassified ≥ MinExcludedNeg Then
                                                                                   (step 5)
   If ChildRule.Score > BestRule.Score Then                                        (step 6)
     BestRule = ChildRule
   If ChildRule.Consistency ≠ 100% Then                                            (step 7)
     ChildRuleSet = ChildRuleSet ∪ {ChildRule}                                     (step 8)
   Else
     Mark $\text{Condition}_i$ as Invalid for ParentRule                           (step 9)
 Else
   Mark $\text{Condition}_i$ as Invalid for ParentRule                             (step 10)
```

Figure 3.4 Modified Search Space Pruning Control Structure for RULES-7

specific as or more specific than the *ChildRule* already formed. Such rules are obviously not needed as a more general consistent rule already exists.

The advantages gained with the modified control structure are as follows:

- No unwanted rules in the final rule set
- Rules in the final rule set truly representative of the regularities within the dataset
- Reduced execution time

Tests conducted in section 3.5.1 on a total of 7 datasets using different values of MinPositives and MinExcludedNeg constraints indicate that the proposed control structure modification incorporated into the RULES-7 algorithm results in a greater noise immunity together with some reduction in the execution time. The user-specified MinPositives criterion now achieves its real intended objective by ensuring that the final rule set includes only rules that satisfy this criterion and consequently are not based on noise in the data. The MinExcludedNeg criterion also is properly applied to improve the discrimination ability of the rules. The final rule set is more compact and the included rules are truly representative of the regularities contained within the dataset and as a result perform very well on unseen data.

3.3.2.2 Other Efficiency Improvements

The execution speed of the algorithm has also been improved by streamlining the way InvalidValues are handled. In RULES-6, there are two arrays for storing the ValidValues and InvalidValues of every single rule. Later, the algorithm subtracts the InvalidValues set of a *ParentRule* from the ValidValues set of a *ChildRule* in order to obtain its ValidValues set for subsequent iterations. This is cumbersome and computationally intensive.

Instead, RULES-7 adopts the practice in RULES-3 Plus, which copies all the attribute-value pairs of the seed instance to the most general rule (*BestRule*) and marks them as "not available" for this rule. Because of this, the attribute-value pairs are already stored in memory for every rule and a simple flag mechanism is used to decide whether an attribute-value pair is included in a rule or not. RULES-7 exploits the same flag mechanism to "mark" the attribute-value pairs of a *ParentRule* as Invalid and then later mark the corresponding attribute-value pairs of its remaining *ChildRules* as Invalid. This does not require maintaining two arrays or performing any additions and subtractions, which leads to faster execution.

3.3.3 Specialisation Measure Parameter

The RULES-6 algorithm uses the m-probability-estimate (Cestnik, 1990) expressed by equation 2.13 as the specialisation measure but sets $m = k$ as in a Laplacian Error Estimate represented by equation 2.10. However, m does not have to be equal to k. The parameter m in the m-probability-estimate tells the algorithm about the amount of noise in the data and is a domain-dependent parameter (Afify, 2004). m is unrelated to k, the number of classes, and should also be specified by the user in order to improve the quality of the induced rule set. If m is set to k then $P_0(C_t)$ should also be set equal to $1/k$, in which case the specialisation measure becomes the Laplacian Error Estimate. Setting $m = k$ for a dataset having many classes but little noise will result in an over-general rule set, which is not the intended purpose of the noise control parameter. Consequently, RULES-7 uses the original m-probability-estimate, i.e. without substituting $m = k$. The default value of m is set to 2, a value empirically found to give good noise control.

3.4 The RULES-7 Algorithm

A pseudo-code description of the RULES-7 algorithm is given in Figure 3.5 and the *Induce_One_Rule* procedure is outlined in Figure 3.6. Apart from addressing the issues discussed in section 3.3, three new techniques shown at steps 4, 7, and 9 have been deployed in the RULES-7 algorithm. The techniques are equally applicable to the entire RULES *family* of algorithms as well as to any covering algorithm in general. The first of the proposed techniques is based on a new *comparison heuristic* in order to boost the classification accuracy of the rule set. The second one is a *ParentRule pruning technique* based on the *full database support* (MinSupport), and the third one is a *ChildRule pruning technique* based on the *conditional database support* (MinPosSupport). The three new techniques are discussed in detail in sections 3.4.1 – 3.4.3.

3.4.1 The Comparison Heuristic (λ)

The RULES-6 algorithm employs the m-probability-estimate as a specialisation measure in order to compute the score of the *ChildRule* and then compares it with the score of the last *BestRule* using the following step:

```
If ChildRule.Score > BestRule.Score Then
   BestRule = ChildRule
```

The comparison step favours the ChildRule over the last BestRule based on this score (equation 2.13), which takes into account only the "Classified" and "Covered" instances of the *ChildRule* to compute its quality, taking no account of the "New Classified" instances covered by the *ChildRule* as compared with the last *BestRule*. This failure to take into

```
Procedure RULES-7 (Dataset, $w$, $\lambda$, MinSup, MinPosSup)
RuleSet = $\emptyset$                                                        (step 1)
Compute the no. of instances a ParentRule should cover as per MinSup         (step 2)
While there are instances in the Dataset that have not been covered Do       (step 3)
  Take a seed example $s$ that has not yet been covered.                     (step 4)
  Rule = Induce_One_Rule ($s$, DataSet, $w$, $\lambda$, MinSup, MinPosSup)   (step 5)
  Mark the instances covered by the Rule as covered.                         (step 6)
  Add Rule to the RuleSet                                                    (step 7)
End While
Return RuleSet                                                               (step 8)
End                                                                          (step 9)
```

Figure 3.5 A pseudo-code description of RULES-7

```
Procedure Induce_One_Rule (s, DataSet, w, $\lambda$, MinSup, MinPosSup)
ParentRuleSet = ChildRuleSet = $\emptyset$
BestRule = rule with no condition (empty antecedent)
Compute the no. of instances a ChildRule should cover within the target class as per
MinPosSup                                                                    (step 1)
Add BestRule to the ParentRuleSet                                            (step 2)
While ParentRuleSet is NOT Empty Do                                          (step 3)
 For each ParentRule $\in$ ParentRuleSet Do
  If ParentRule.Covered $\geq$ MinSupport Then                                (step 4)
   For each nominal Attribute $A_i$ Do
    If Condition$_i$ $\notin$ ParentRule AND Condition$_i$ is not marked as "Invalid" for ParentRule
    Then
      ChildRule = ParentRule $\wedge$ Condition$_i$                               (step 5)
     If ChildRule $\notin$ ChildRuleSet Then                                  (step 6)
     {Check the ChildRuleSet for duplicate rule of the candidate ChildRule}
      Compute ChildRule.Instances
      If ChildRule.Classified $\geq$ MinPosSupport AND                        (step 7)
        ParentRule.Misclassified $-$ ChildRule.Misclassified $\geq$ MinExcludedNeg Then
                                                                             (step 8)
       Compute ChildRule.Quality
       If (ChildRule.Score > BestRule.Score) OR
          (ChildRule.Score > $\lambda \times$ BestRule.Score AND
           ChildRule.NewClassified > BestRule.NewClassified) Then            (step 9)
          BestRule = ChildRule
       If ChildRule.Consistency $\neq$ 100% Then                               (step 10)
         Add ChildRule to ChildRuleSet                                       (step 11)
       Else
         Mark Condition$_i$ as "Invalid" for ParentRule                        (step 12)
      Else
        Mark Condition$_i$ as "Invalid" for ParentRule                         (step 13)
   End For
```

Figure 3.6 A pseudo-code description of the *Induce_One_Rule* procedure of RULES-7

```
    End For
    For each $ChildRule \in ChildRuleSet$ Do
      Compute $ChildRule$.OptimisticScore
      If $ChildRule$.OptimisticScore $\leq$ $BestRule$.Score Then                    (step 14)
        Delete the $ChildRule$ from the $ChildRuleSet$                               (step 15)
        For each nominal Attribute $A_i$ Do
          If Condition$_i \in ChildRule$ AND Condition$_i \notin ParentRule$ Then      (step 16)
            Mark Condition$_i$ as "Invalid" for $ParentRule$                         (step 17)
            Exit For Loop
        End For
    End For
    For each $ChildRule \in ChildRuleSet$ Do
    {Mark any Conditions of remaining $ChildRules$ of each $ParentRule$ as Invalid which are
     "marked as Invalid" for the $ParentRule$}
      For each nominal Attribute $A_i$ Do
        If Condition$_i$ is marked as "Invalid" for $ParentRule$ Then                (step 18)
          Mark Condition$_i$ as "Invalid" for $ChildRule$                            (step 19)
      End For
    End For
    Empty $ParentRuleSet$
    If $w > 1$ Then
      Add $w$ highest Score $ChildRules$ from $ChildRuleSet$ into $ParentRuleSet$    (step 20)
      Empty $ChildRuleSet$
  End While
  Return $BestRule$
  End
```

Figure 3.6 A pseudo-code description of the *Induce_One_Rule* procedure of RULES-7 (continued).

account the "New Classified" instances will logically result in an increased overlapping among the rules, thereby increasing the size of the rule set, which puts a greater overhead on the post-pruning phase to follow. Secondly, since the specificity of the rules is directly proportional to the number of rules, so a higher number of rules will also result in a lower classification accuracy on the test data.

The proposed *comparison heuristic* for the RULES-7 algorithm is based on the following idea:

"If *ChildRule*.Score is greater than *BestRule*.Score, then there are definitely no two opinions and the *ChildRule* should replace the last *BestRule*. If however, *ChildRule*.Score is NOT greater than *BestRule*.Score (in which case it might be equal to or less than *BestRule*.Score), then its competitiveness is assessed with respect to the last *BestRule*. The *ChildRule* is considered a very competitive rule if its score is between 90-99% of *BestRule*.Score AND it covers more *New Positive Instances* (positive instances not covered by previous rules in the rule set) as compared with the last *BestRule*. If these two conditions hold true, then the *ChildRule* still qualifies as an improvement over the last *BestRule* and replaces it."

Based on the above-mentioned idea, the proposed *comparison heuristic* for the RULES-7 algorithm is as follows:

```
If ChildRule.Score > BestRule.Score OR
  (ChildRule.Score > $\lambda \times$ BestRule.Score) AND (ChildRule.NewClassified >
  BestRule.NewClassified) Then
  BestRule = ChildRule
```

where λ = *comparison threshold*

An appropriate empirical range for λ is between 90% - 99%, although 90%, 95% and 99% have proved to be the optimal values for most of the datasets. Tests conducted in section 3.5.2 on a total of 40 datasets using the new *comparison heuristic* prove that the classification accuracy of the induced rule set increases by as much as 10%.

3.4.2 ParentRule Pruning Technique (MS)

The RULES-6 algorithm uses a MinPositives constraint specified by the user to prune only the search space. This is done by checking if the *ChildRule* of any *ParentRule* satisfies the constraint and if not, the last attribute-value used to form that *ChildRule* is added to the InvalidValues set of the *ParentRule* in order to make sure that the remaining fruitful *ChildRules* of that parent also don't use that attribute-value in their specialisation process. As already mentioned, this constraint wasn't able to achieve its real objective of pruning any over-specific *ChildRules* as well as pruning the search space because the *comparison step* was placed outside the *search space pruning* steps. Nevertheless, with the modified control structure proposed in RULES-7, it became possible to exploit the user-specified MinPositives constraint fully so as to achieve the desired objective, which is to increase noise immunity as well as to cut down the search space.

There is however, yet another top-level user-specified constraint not exploited in the RULES-6 algorithm that can be used to drastically cut down the search space in the case of large noisy datasets, the MinSupport constraint. This constraint has been used successfully in association rule mining algorithms in order to limit the search to patterns of interest to the end-user. The MinPositives constraint used in RULES-6 prunes only those rules that do not satisfy the user-specified support within the *target class*. The proposed support-based constraint also prunes rules that do not satisfy the user-specified minimum support within the *whole database*. However, the effectiveness of the MinSupport constraint is far greater than the MinPositives constraint as far as cutting down the search space is concerned. This is because the MinSupport constraint can be used to prune an entire *ParentRule* instead of first generating its *ChildRules* and then pruning them. The format of the MinSupport constraint is as follows:

```
If ParentRule.Covered ≥ MinSupport Then                    (step 4)
...
  ChildRule = ParentRule ∧ Condition_i                     (step 5)
```

The usefulness of the MinSupport constraint can be illustrated as follows:

IF *Outlook* = *sunny* **THEN** *Class* = *yes* *ParentRule*

IF *Outlook* = *sunny* **AND** *Humidity* = *normal* **THEN** *Class* = *yes* *ChildRule*

Suppose we take both MinPositives ≥ 3 and MinSupport ≥ 3.

The constraint MinPositives ≥ 3 prunes the *ChildRule* if it covers less than 3 instances within the target class (which is the conditional database, i.e. given the condition *Class* = *yes*). In other words, if the pattern "*sunny*, *normal*, *yes*" appears less than 3 times within the whole database, then, as per the user-specified MinPositives criterion, this pattern does not qualify for further specialisation and is therefore pruned.

The constraint MinSupport ≥ 3 however is used at the top-level and prunes the *ParentRule* if its coverage (i.e. the support of its antecedent *Outlook* = *sunny*) is less than 3 within the whole database. In other words, if the pattern "*sunny*" appears less than 3 times within the whole database, then the rule having this pattern in its antecedent is useless as per user-specified MinSupport criterion. Consequently, it does not qualify for further specialisation and is therefore pruned right away without even considering the class condition. This is because appending the class condition (*Class* = *yes*) to the pattern "*sunny*" results in the pattern "*sunny*, *yes*", which is only a special case of the pattern "*sunny*" and so its support will obviously be less than 3. Therefore it will also not qualify for further specialisation and will be pruned.

The support-based pruning constraint MinSupport is therefore far more effective than the MinPositives constraint which is only used at the sub-level, i.e. for pruning *ChildRules* of a *ParentRule*. This is because the MinSupport constraint prunes an entire *ParentRule* instead of going down to the specialisation phase to first produce its *ChildRules* and then prune them using the MinPositives constraint. As a result, the constraint can drastically cut down the search space in the case of large noisy datasets even though it does not flag any Conditions as invalid.

The MinSupport constraint is always taken as a percentage of the total number of instances in the database, with a general range between 5% - 95%. Tests conducted in section 3.5.3 on a total of 40 datasets employing the new constraint MinSupport prove that the length of the rule set reduces by as much as 93%, the accuracy increases by as much as 23%, and the learning time also reduces by as much as 97%.

3.4.3 ChildRule Pruning Technique (MPS)

The user-specified MinPositives pruning technique employed in the RULES-6 algorithm suffers from an inherent drawback. The limitation lies in using a static support for all the member classes within the database. Since the class distribution within the database is almost always not uniform, setting a static support for all the member classes results in an inappropriate percentage to be satisfied by the *ChildRules* of some classes. This can be illustrated as follows.

Suppose there are 3 classes within a dataset, class 1, class 2 and class 3 and the number of instances in each class is 4, 6 and 10 respectively. Setting MinPositives = 5 will result in skipping class 1 altogether because class 1 has only 4 instances in it and therefore all *ChildRules* of any *ParentRule* with class 1 as the consequent will simply be pruned. Similarly with MinPositives = 5, *ChildRules* of class 2 will have to satisfy 83.33% support whereas *ChildRules* of class 3 will have to satisfy only 50% support. So it is clearly evident that the user-specified static MinPositives support does not take into account the class distribution within the dataset. The new *ChildRule* pruning technique overcomes this limitation by introducing the idea of dynamic support for every member class within the dataset.

To handle the problem of minimum support for classes having asymmetrical distribution, the HARMONY algorithm (Wang and Karypis, 2006) provides two options to the user to specify class-specific minimum supports. The first option allows the user to specify a minimum support for each class, which is obviously very cumbersome in the case that there are many classes within the dataset. The second option requires the user to specify a minimum support for the smallest class, and the support for the remaining classes is calculated using the following equation:

$$min\_sup_i = min\_sup \times \left(\frac{|c_i|}{\min_{\forall j,\ 1 \leq j \leq k} |c_j|} \right) \xi \qquad (3.1)$$

where min_sup = minimum support for the smallest class specified by the user

min_sup_i = minimum support for the new class

$\min_{\forall j,\ 1 \leq j \leq k} |c_j|$ = number of instances in the smallest class

ξ = support differentia factor, ≥ 0

So if, for example, the user specifies a minimum support of 2 for the smallest class, i.e. class 1 with 4 instances, then the minimum supports for class 2 and class 3 having 6 and 10 instances respectively turn out to be 3 and 5 respectively according to equation 3.1. However, a more straightforward way of doing this is to use a support percentage just like the MinSupport constraint, which has to be satisfied by the *ParentRule* with respect to the whole database. The MinPositives constraint specified by the user in the form of a percentage is referred to as MinPosSupport (conditional database support) in RULES-7 and should be satisfied by all *ChildRules* of any *ParentRule* in order to qualify for further specialisation. In this case, if MinPosSupport = 50% then the minimum supports for class 1, class 2 and class 3 turn out to be 2, 3 and 5 respectively, exactly as predicted by equation 3.1. If the *ChildRule* does not satisfy the MinPosSupport constraint, then the last Condition used to specialise it is flagged as invalid, so that it is not used to specialise other *ChildRules* of the same *ParentRule*.

The MinPosSupport constraint is taken as a percentage of the total number of instances within the target class, with a general range between 5% - 95%. Tests conducted in section 3.5.4 on a total of 40 datasets employing the new constraint MinPosSupport prove that the length of the rule set reduces by as much as 93%, the accuracy increases by as much as 27%, and the learning time also reduces by as much as 100%.

3.5 Empirical Evaluation of RULES-7

This section presents the results obtained from experimental evaluation of RULES-7 against RULES-6. For comprehensive comparison between the two algorithms, experiments were conducted on 40 benchmark datasets, all of them obtained from the University of California at Irvine (UCI) repository of machine learning databases (Blake and Merz, 1998), except the *Depression* dataset obtained from Williams College (Veaux, 2007). Table 3.3 presents a summary of these datasets, which are a mix of nominal, continuous and mixed type data. Each dataset name is followed by either (N), (C) or (M) indicating the type of the dataset, and occasionally by the suffix L indicating that the dataset is large. This study considers a dataset large if the product of the number of instances and the number of attributes within the dataset is greater than 50,000. Because of their extremely large size, the datasets *Connect-4(C)L* and *Cover-Type(M)L* could not be mined by RULES-6 within a reasonable amount of time, although they could be handled quite easily by RULES-7 using the new support-based pruning techniques. Consequently, these datasets were sampled to approximately 1/20 and 1/187 of their full datasets containing 67,557 and 581,012 instances respectively in order to facilitate comparison with RULES-6.

All tests were conducted on an Intel Pentium 2.0 GHz Dual-Core computer with 2 GB of RAM and Windows XP operating system. For continuous and mixed type datasets, the discretisation was carried out using the equal-width approach (Wong and Chiu, 1987), with the number of intervals set to 6. The evaluation approach used in RULES-6 was as follows. For datasets with more than 1000 instances, each set was randomly divided once into a

No.	Dataset	No. of Instances	No. of Attributes			No. of Classes (k)
			Total	Nominal	Continuous	
1	Adult(M)L	48,842	14	8	6	2
2	Anneal(M)	798	38	32	6	6
3	Arrhythmia(M)L	452	279	73	206	13
4	Breast-Cancer(C)	699	10	0	10	2
5	Breast-Cancer(N)	286	9	9	0	2
6	Car(N)	1,728	6	6	0	4
7	Chess(N)L	3,196	36	36	0	2
8	*Connect-4(N)L	3,376	42	42	0	2
9	*Cover-Type(M)L	3,100	54	47	7	7
10	Credit-Approval(M)	690	15	9	6	2
11	Depression(M)	428	17	11	6	2
12	Dermatology(M)	366	34	33	1	6
13	Ecoli(C)	336	8	0	8	8
14	Flags(M)	194	29	29	0	8
15	German-Credit(M)	1,000	20	13	7	2
16	Hayes-Roth(N)	160	5	5	0	2
17	Heart-Cleveland(M)	303	13	8	5	5
18	Heart-Hungarian(M)	294	13	8	5	2
19	Hepatitis(M)	155	19	13	6	2
20	Horse-Colic(M)	368	27	18	9	2
21	Hyperthyroid(M)L	3,711	29	22	7	4
22	Hypothyroid(M)L	3,772	29	22	7	2
23	Image(C)	210	19	0	19	7
24	Ionosphere(C)	351	34	0	34	2
25	Iris(C)	150	4	0	4	3
26	Lymphography(N)	148	18	18	0	4
27	Mushroom(N)L	8,124	22	22	0	2
28	Nursery(N)L	12,960	8	8	0	2
29	Parkinsons(C)	195	22	22	0	2
30	Pendigits(C)L	10,992	16	0	16	10
31	Pima-Indians(C)	768	8	8	0	2
32	Post-Operative-Patient(M)	90	8	8	0	3
33	Promoters(N)	106	58	58	0	2
34	Soybean-Large(N)	683	35	35	0	19
35	Spect(C)	267	44	0	44	2
36	SPECT-Heart(N)	267	22	22	0	2
37	Splice(N)L	3,190	61	61	0	3
38	Tic-Tac-Toe(N)	958	9	9	0	2
39	Vehicle(C)	846	18	0	18	4
40	Wine(C)	178	13	0	13	3

Table 3.3 Summary of Datasets used in Experiments

training set with two-thirds of the data and a test set with the remaining one-third. For datasets with fewer than 1000 instances, this procedure was repeated ten times and the results were averaged (Pham and Afify, 2005b). The evaluation approach was therefore partly threefold and partly tenfold and also did not indicate whether stratification was carried out or not. By contrast, in order to ensure the accuracy of the prediction estimate, the approach used throughout this study was the standard *stratified 10-fold cross-validation* (Kohavi, 1995a, Witten and Frank, 2005). To evaluate the performance of the algorithms, three criteria were used, namely *Rules_Reduction*, *Accuracy_Increase* and *Time_Reduction*, of which the second criterion is of prime importance in most induction tasks. A dataset satisfies the *Overall Improvement Criterion* if one of the above-mentioned three criteria is equal to zero and the other two are greater than zero. All execution times are reported in seconds.

Section 3.5.1 compares the results of the RULES-7 algorithm with the modifications suggested in section 3.3 and without any pruning technique against those of RULES-6. For both the algorithms, the parameters *beam width*, *MinPositives* and *MinExcludedNeg* were set to 4, 2 and 1 respectively. Sections 3.5.2 – 3.5.4 compare the results obtained by employing the three new techniques, namely *Comparison Threshold*, *MinSupport* and *MinPosSupport* respectively in the RULES-7 algorithm against those of RULES-6. Finally, section 3.5.5 compares the results presented in the previous three sections.

3.5.1 RULES-7 vs. RULES-6

Table 3.4 shows the effect of beam width on the execution time of RULES-7 as compared with that of RULES-6 on a total of 7 datasets. Furthermore, Table 3.5 and Table 3.6 use the same datasets in order to illustrate the effect of varying the values of MinPositives and

No.	Dataset	w ∈ {4, 6, 8, 10, 12}	No. of Rules		Training Accuracy (%)		Exec. Time (sec)		Redc. (%)
			RULES6	RULES7	RULES6	RULES7	RULES6	RULES7	
1	Breast-Cancer(N)	4	73	72	93.57	93.57	0.25	0.21	**16.00**
		6	74	73	94.07	94.07	0.37	0.29	**21.62**
		8	72	72	94.07	94.07	0.49	0.35	**28.57**
		10	73	72	94.11	94.11	0.58	0.40	**31.03**
		12	73	72	94.03	94.07	0.66	0.46	**30.30**
2	Depression(M)	4	120	118	99.22	99.22	1.15	1.08	**6.09**
		6	117	117	99.38	99.35	1.67	1.54	**7.78**
		8	117	117	99.43	99.40	2.77	1.92	**30.69**
		10	116	116	99.46	99.46	2.82	2.31	**18.09**
		12	114	114	99.46	99.46	3.38	2.65	**21.60**
3	German-Credit(M)	4	239	239	99.52	99.53	7.12	6.72	**5.62**
		6	237	237	99.66	99.64	10.48	9.53	**9.06**
		8	235	236	99.70	99.71	13.91	12.69	**8.77**
		10	235	235	99.73	99.74	17.62	15.57	**11.63**
		12	234	234	99.77	99.73	21.41	18.86	**11.91**
4	Image(C)	4	38	37	95.34	95.19	0.17	0.16	**5.88**
		6	37	37	95.40	95.34	0.27	0.23	**14.81**
		8	37	37	95.56	95.61	0.36	0.30	**16.67**
		10	38	37	95.77	95.77	0.45	0.38	**15.56**
		12	37	37	95.82	95.82	0.56	0.45	**19.64**
5	P-O-Patient(M)	4	25	25	83.78	86.46	0.03	0.02	**33.33**
		6	25	25	83.66	86.46	0.04	0.03	**25.00**
		8	25	25	83.66	86.59	0.05	0.03	**40.00**
		10	25	25	83.66	86.83	0.05	0.04	**20.00**
		12	25	25	83.66	86.83	0.06	0.05	**16.67**
6	Soybean-Large(N)	4	55	54	97.56	97.58	1.82	1.73	**4.95**
		6	55	54	97.77	97.75	2.77	2.59	**6.50**
		8	55	55	97.77	97.77	3.70	3.47	**6.22**
		10	55	55	97.87	97.87	4.67	4.32	**7.49**
		12	55	55	98.01	98.01	5.71	5.19	**9.11**
7	Vehicle(C)	4	198	197	95.38	95.33	4.41	4.14	**6.12**
		6	196	196	95.72	95.73	6.40	5.82	**9.06**
		8	194	194	95.63	95.58	8.41	7.36	**12.49**
		10	196	195	95.72	95.72	10.66	8.96	**15.95**
		12	194	193	95.79	95.76	12.99	10.55	**18.78**

Table 3.4 RULES-7 vs. RULES-6 with respect to Beam Width

No.	Dataset	Min Positives ∈ {1, 2, 3, 4, 5}	No. of Rules		Accuracy (%)		Incr. (%)	Exec. Time (sec)		Redc. (%)
			RULES6	RULES7	RULES6	RULES7		RULES6	RULES7	
1	Breast-Cancer(N)	1	74	74	66.43	66.07	**-0.54**	0.25	0.22	**12.00**
		2	73	72	66.07	66.79	**1.09**	0.25	0.22	**12.00**
		3	73	70	68.21	68.57	**0.53**	0.24	0.21	**12.50**
		4	71	65	68.21	68.79	**0.85**	0.24	0.19	**20.83**
		5	68	62	66.43	66.83	**0.60**	0.23	0.18	**21.74**
2	Depression(M)	1	119	119	66.67	65.71	**-1.44**	1.17	1.10	**5.98**
		2	120	118	66.43	65.48	**-1.43**	1.15	1.09	**5.22**
		3	119	118	67.38	66.90	**-0.71**	1.15	1.08	**6.09**
		4	117	114	66.67	67.62	**1.42**	1.13	1.03	**8.85**
		5	115	109	67.38	67.90	**0.77**	1.08	0.97	**10.19**
3	German-Credit(M)	1	239	240	71.90	71.70	**-0.28**	7.13	6.72	**5.75**
		2	239	239	71.90	72.40	**0.70**	7.12	6.72	**5.62**
		3	239	239	71.60	71.20	**-0.56**	7.11	6.83	**3.94**
		4	239	238	71.80	72.50	**0.97**	7.10	6.67	**6.06**
		5	238	235	71.60	72.10	**0.70**	7.06	6.58	**6.80**
4	Image(C)	1	37	37	76.67	80.00	**4.34**	0.17	0.16	**5.88**
		2	38	37	76.67	80.48	**4.97**	0.17	0.16	**5.88**
		3	37	36	75.71	80.00	**5.67**	0.18	0.16	**11.11**
		4	36	34	76.67	81.90	**6.82**	0.16	0.13	**18.75**
		5	34	32	76.19	80.95	**6.25**	0.14	0.12	**14.29**
5	P-O-Patient(M)	1	31	31	65.00	64.50	**-0.77**	0.03	0.03	**0.00**
		2	25	25	65.00	68.75	**5.77**	0.02	0.02	**0.00**
		3	25	19	63.75	65.03	**2.01**	0.03	0.02	**33.33**
		4	25	18	63.75	64.97	**1.91**	0.02	0.02	**0.00**
		5	25	18	65.00	66.50	**2.31**	0.03	0.01	**66.67**
6	Soybean-Large(N)	1	54	54	89.84	90.16	**0.36**	1.79	1.75	**2.23**
		2	55	54	90.00	90.31	**0.34**	1.93	1.73	**10.36**
		3	54	54	90.00	89.70	**-0.33**	1.75	1.72	**1.71**
		4	54	54	90.00	89.50	**-0.56**	1.78	1.72	**3.37**
		5	53	53	90.63	91.25	**0.68**	1.75	1.70	**2.86**
7	Vehicle(C)	1	198	197	66.83	67.44	**0.91**	4.38	4.09	**6.62**
		2	198	197	67.20	67.56	**0.54**	4.41	4.14	**6.12**
		3	197	195	67.20	68.17	**1.44**	4.39	4.09	**6.83**
		4	194	186	66.59	67.68	**1.64**	4.29	3.86	**10.02**
		5	189	174	68.05	69.27	**1.79**	4.09	3.55	**13.20**

Table 3.5 RULES-7 vs. RULES-6 with respect to MinPositives

No.	Dataset	Min Negatives ∈ {1, 2, 3, 4, 5}	No. of Rules		Accuracy (%)		Incr. (%)	Exec. Time (sec)		Redc. (%)
			RULES6	RULES7	RULES6	RULES7		RULES6	RULES7	
1	Breast-Cancer(N)	1	73	72	66.07	66.79	**1.09**	0.25	0.21	**16.00**
		2	74	72	66.79	68.21	**2.13**	0.25	0.21	**16.00**
		3	73	68	67.50	69.29	**2.65**	0.23	0.20	**13.04**
		4	72	64	67.50	70.00	**3.70**	0.22	0.18	**18.18**
		5	71	63	68.21	71.43	**4.72**	0.21	0.18	**14.29**
2	Depression(M)	1	120	118	66.43	65.48	**-1.43**	1.15	1.09	**5.22**
		2	118	118	67.14	66.43	**-1.06**	1.09	1.02	**6.42**
		3	117	109	67.62	66.43	**-1.76**	1.02	0.91	**10.78**
		4	116	98	68.33	67.38	**-1.39**	0.97	0.78	**19.59**
		5	117	90	69.05	65.71	**-4.84**	0.95	0.70	**26.32**
3	German-Credit(M)	1	239	239	71.90	72.40	**0.70**	7.12	6.72	**5.62**
		2	241	241	71.80	71.70	**-0.14**	6.83	6.48	**5.12**
		3	243	235	71.10	71.70	**0.84**	6.73	6.22	**7.58**
		4	242	225	71.50	71.70	**0.28**	6.48	5.83	**10.03**
		5	244	217	72.20	72.30	**0.14**	6.35	5.47	**13.86**
4	Image(C)	1	38	37	76.67	80.48	**4.97**	0.17	0.16	**5.88**
		2	37	37	75.24	79.52	**5.69**	0.15	0.14	**6.67**
		3	37	36	76.19	81.90	**7.49**	0.13	0.12	**7.69**
		4	37	37	75.71	80.95	**6.92**	0.13	0.12	**7.69**
		5	37	36	77.14	81.90	**6.17**	0.12	0.11	**8.33**
5	P-O-Patient(M)	1	25	25	65.00	68.75	**5.77**	0.03	0.02	**33.33**
		2	25	24	65.00	70.00	**7.69**	0.02	0.02	**0.00**
		3	24	21	62.50	67.50	**8.00**	0.02	0.02	**0.00**
		4	24	19	61.25	66.25	**8.16**	0.02	0.01	**50.00**
		5	23	18	62.50	68.75	**10.00**	0.02	0.01	**50.00**
6	Soybean-Large(N)	1	55	54	90.00	90.31	**0.34**	1.77	1.73	**2.26**
		2	55	53	90.00	90.31	**0.34**	1.74	1.64	**5.75**
		3	55	55	89.69	91.25	**1.74**	1.67	1.63	**2.40**
		4	54	54	89.53	89.69	**0.18**	1.61	1.54	**4.35**
		5	54	53	89.38	89.06	**-0.36**	1.56	1.48	**5.13**
7	Vehicle(C)	1	198	197	67.20	67.56	**0.54**	4.41	4.14	**6.12**
		2	195	192	66.46	67.93	**2.21**	4.16	3.90	**6.25**
		3	198	189	65.37	65.61	**0.37**	4.19	3.75	**10.50**
		4	195	182	65.49	67.93	**3.73**	3.99	3.59	**10.03**
		5	193	175	66.22	69.27	**4.61**	3.90	3.38	**13.33**

Table 3.6 RULES-7 vs. RULES-6 with respect to MinExcludedNeg

MinExcludedNeg respectively on the performance of RULES-7 as compared with RULES-6. Table 3.7 compares the results obtained from RULES-7 without using any new technique against those of RULES-6 on the 40 datasets mentioned in Table 3.3. A summary of the results for RULES-7 is presented in tabular form in Table 3.8, as well as in graphical form in Figures 3.7 – 3.9. The last row in Table 3.7 displays the total for all the datasets, from which it can be seen that the cumulative number of rules for RULES-7 stays more or less the same. However, the cumulative accuracy increases by 1.01% and the learning time decreases by 11.02%. The increase in accuracy can be attributed solely to the control structure modification suggested in section 3.3.2 whereas the reduction in the learning time is primarily due to the duplicate rules correction step in section 3.3.1. The most notable increase in accuracy (greater than 1.5%) can be seen for 12 datasets listed as numbers 3, 6, 13, 16, 23, 26, 27, 29, 31, 32, 33 and 40. The top 5 datasets in terms of an accuracy boost are *Promoters(N), Arrhythmia(M)L, P-O-Patient(M), Image(C), Parkinsons(C)* with an accuracy boost of 7.69%, 5.88%, 5.77%, 4.97% and 3.70% respectively.

It is also evident from Table 3.8 that 58% of the datasets satisfy the *overall improvement criterion*. The classification accuracy for 17% of the datasets is the same for both the algorithms. For 70% of the datasets, the accuracy achieved by RULES-7 is higher whereas for the remaining 13% it is lower. The number of rules induced by RULES-7 is lower for 50% and higher for 10% of the datasets, whereas the learning time for RULES-7 is lower for 73% and higher for only 2% of the datasets. These results clearly indicate that even without the new techniques, the RULES-7 algorithm is more accurate and scalable than RULES-6.

No.	Dataset	No. of Rules		Redc. (%)	Accuracy (%)		Incr. (%)	Exec. Time		Redc. (%)	Speed Boost (*x*)	Impr. ?
		Rules6	Rules7		Rules6	Rules7		Rules6	Rules7			
1	Adult(M)L	191	188	**1.57**	77.90	77.91	**0.01**	400.45	396.67	**0.94**	1.0	yes
2	Anneal(M)	40	40	**0.00**	97.79	97.79	**0.00**	0.52	0.45	**13.46**	1.2	no
3	Arrhythmia(M)L	110	110	**0.00**	55.25	58.50	**5.88**	82.43	87.33	**-5.94**	nil	no
4	Breast-Cancer(C)	27	25	**7.41**	94.64	94.64	**0.00**	0.09	0.09	**0.00**	nil	no
5	Breast-Cancer(N)	73	72	**1.37**	66.07	66.79	**1.09**	0.25	0.22	**12.00**	1.1	yes
6	Car(N)	94	94	**0.00**	60.64	61.70	**1.75**	0.97	0.73	**24.74**	1.3	yes
7	Chess(N)L	47	48	**-2.13**	94.06	94.21	**0.16**	22.58	17.37	**23.07**	1.3	no
8	Connect-4(N)L	401	400	**0.25**	66.38	66.74	**0.54**	191.58	129.89	**32.20**	1.5	yes
9	Cover-Type(M)L	448	449	**-0.22**	56.22	56.42	**0.36**	157.06	127.43	**18.87**	1.2	no
10	Credit-Approval(M)	72	72	**0.00**	80.44	81.03	**0.73**	1.28	1.19	**7.03**	1.1	yes
11	Depression(M)	120	118	**1.67**	66.43	65.48	**-1.43**	1.16	1.10	**5.17**	1.1	no
12	Dermatology(M)	35	36	**-2.86**	88.29	88.57	**0.32**	0.82	0.70	**14.63**	1.2	no
13	Ecoli(C)	50	48	**4.00**	78.39	80.00	**2.05**	0.10	0.08	**20.00**	1.3	yes
14	Flags(M)	57	57	**0.00**	58.13	58.75	**1.07**	0.57	0.47	**17.54**	1.2	yes
15	German-Credit(M)	239	239	**0.00**	71.90	72.40	**0.70**	7.13	6.71	**5.89**	1.1	yes
16	Hayes-Roth(N)	30	18	**40.00**	70.00	72.00	**2.86**	0.02	0.01	**50.00**	2.0	yes
17	Heart-Cleveland(M)	97	97	**0.00**	55.36	55.36	**0.00**	0.55	0.50	**9.09**	1.1	no
18	Heart-Hungarian(M)	48	48	**0.00**	76.79	77.50	**0.92**	0.19	0.18	**5.26**	1.1	yes
19	Hepatitis(M)	30	30	**0.00**	82.00	80.00	**-2.44**	0.11	0.11	**0.00**	nil	no
20	Horse-Colic(M)	69	69	**0.00**	80.83	79.72	**-1.37**	0.64	0.60	**6.25**	1.1	no
21	Hyperthyroid(M)L	72	68	**5.56**	97.99	97.99	**0.00**	25.47	21.44	**15.82**	1.2	yes
22	Hypothyroid(M)L	65	65	**0.00**	90.79	90.85	**0.07**	13.00	11.22	**13.69**	1.2	yes
23	Image(C)	38	37	**2.63**	76.67	80.48	**4.97**	0.17	0.17	**0.00**	nil	yes
24	Ionosphere(C)	49	48	**2.04**	87.06	87.35	**0.33**	0.53	0.53	**0.00**	nil	yes
25	Iris(C)	11	10	**9.09**	95.33	93.33	**-2.10**	0.00	0.00	**0.00**	nil	no
26	Lymphography(N)	30	31	**-3.33**	82.14	84.29	**2.62**	0.11	0.11	**0.00**	nil	no
27	Mushroom(N)L	26	26	**0.00**	97.58	99.56	**2.03**	10.05	8.91	**11.34**	1.1	yes
28	Nursery(N)L	358	357	**0.28**	67.99	65.38	**-3.84**	118.36	90.92	**23.18**	1.3	no
29	Parkinsons(C)	30	29	**3.33**	90.00	93.33	**3.70**	0.13	0.13	**0.00**	nil	yes
30	Pendigits(C)L	500	495	**1.00**	93.17	94.53	**1.46**	248.15	234.61	**5.46**	1.1	yes
31	Pima-Indians(C)	120	100	**16.67**	69.08	70.92	**2.66**	0.96	0.67	**30.21**	1.4	yes
32	P-O-Patient(M)	25	25	**0.00**	65.00	68.75	**5.77**	0.03	0.03	**0.00**	nil	no
33	Promoters(N)	20	20	**0.00**	78.00	84.00	**7.69**	0.15	0.15	**0.00**	nil	no
34	Soybean-Large(N)	55	54	**1.82**	90.00	90.31	**0.34**	1.97	1.77	**10.15**	1.1	yes
35	Spect(C)	54	53	**1.85**	78.46	78.46	**0.00**	1.04	1.04	**0.00**	nil	no
36	SPECT-Heart(N)	34	34	**0.00**	82.69	82.69	**0.00**	0.51	0.46	**9.80**	1.1	no
37	Splice(N)L	254	254	**0.00**	90.32	90.35	**0.03**	173.57	157.26	**9.40**	1.1	yes
38	Tic-Tac-Toe(N)	28	27	**3.57**	92.74	92.74	**0.00**	0.26	0.23	**11.54**	1.1	yes
39	Vehicle(C)	198	197	**0.51**	67.20	67.56	**0.54**	4.38	4.13	**5.71**	1.1	yes
40	Wine(C)	29	28	**3.45**	87.50	90.63	**3.58**	0.05	0.04	**20.00**	1.3	yes
Average:		107	105	**1.36**	78.93	79.73	**1.01**	36.68	32.64	**11.02**		

Table 3.7 RULES-7 vs. RULES-6

No.	Criterion	No. of Datasets for which					
		Redc.		Equal		Incr.	
		Total	%age	Total	%age	Total	%age
1	No. of Rules	20	50%	16	40%	4	10%
2	Accuracy	5	13%	7	17%	28	70%
3	Exec. Time	29	73%	10	25%	1	2%
4	Overall Impr.	23					58%

Table 3.8 Summary of RULES-7

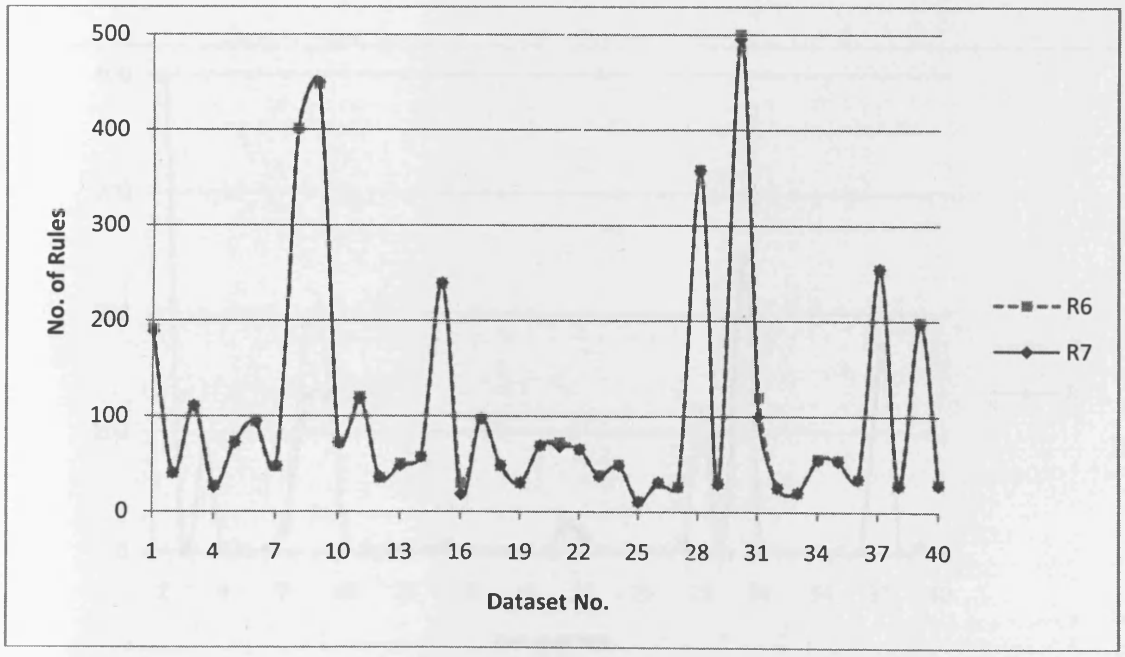


Figure 3.7 RULES-7 vs. RULES-6 in terms of Number of Rules

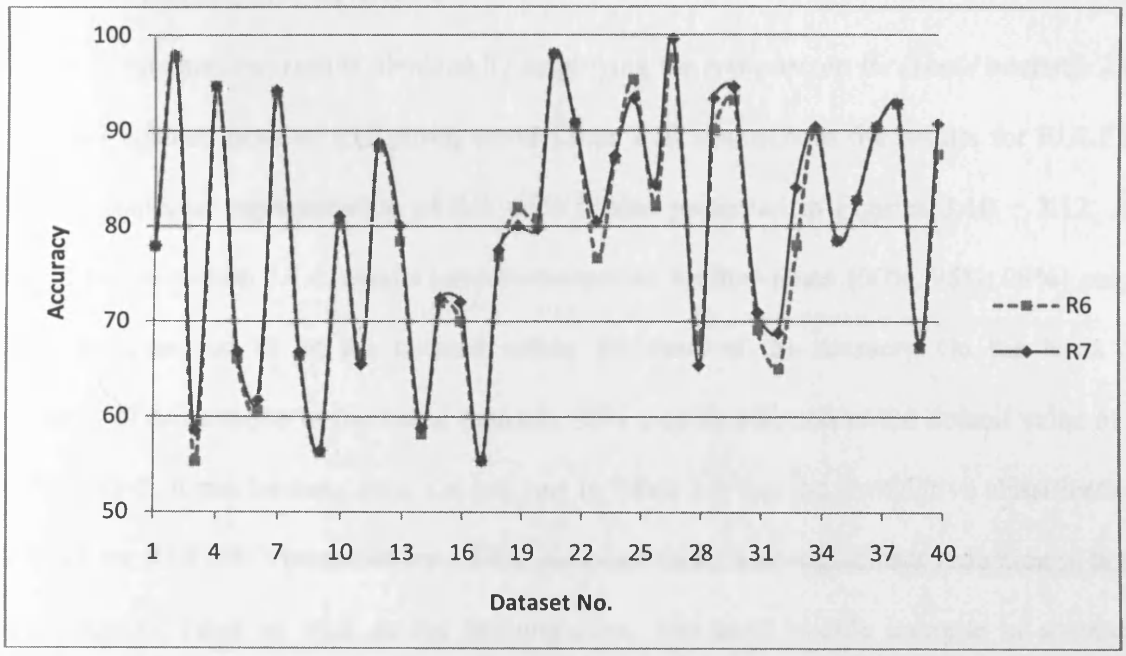


Figure 3.8 RULES-7 vs. RULES-6 in terms of Accuracy

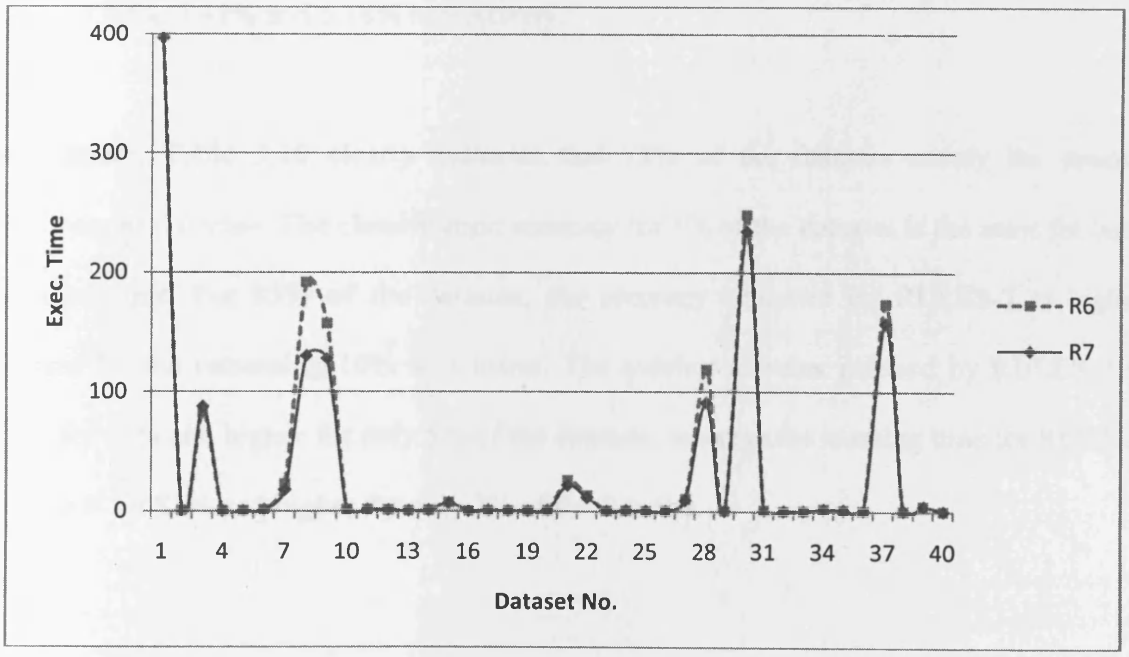


Figure 3.9 RULES-7 vs. RULES-6 in terms of Exec. Time

3.5.2 RULES-7@λ vs. RULES-6

Table 3.9 compares the results obtained by employing the *comparison threshold* heuristic λ in RULES-7 against those of RULES-6, while Table 3.10 summarises the results for RULES-7@λ. A graphical representation of this table is also presented in Figures 3.10 – 3.12. As pointed out in section 3.4.1, results have been reported for the values {90%, 95%, 99%} only, which have proved to be the optimal values for most of the datasets. On the basis of frequency of occurrence in the tested datasets, 99% may be selected as the default value of λ for RULES-7. It can be seen from the last row in Table 3.9 that the cumulative classification accuracy for RULES-7 increases by 2.10%, although there is no significant reduction in both the number of rules as well as the learning time. The most visible increase in accuracy (greater than 1.5%) can be seen for 20 datasets listed as numbers 3, 5, 6, 8, 9, 13, 14-16, 22, 23, 26-29, 31-34 and 38. The top 5 datasets in terms of an accuracy boost are *Promoters(N), Arrhythmia(M)L, P-O-Patient(M), Image(C)* and *Car(N)* with an accuracy boost of 10.26%, 8.60%, 7.69%, 7.45% and 6.18% respectively.

Furthermore, Table 3.10 clearly indicates that 78% of the datasets satisfy the *overall improvement criterion*. The classification accuracy for 5% of the datasets is the same for both the algorithms. For 85% of the datasets, the accuracy achieved by RULES-7 is higher whereas for the remaining 10% it is lower. The number of rules induced by RULES-7 is lower for 95% and higher for only 5% of the datasets, whereas the learning time for RULES-7 is lower for 83% and higher for only 7% of the datasets.

No.	Dataset	λ ∈ {90, 95, 99} %	No. of Rules		Redc. (%)	Accuracy (%)		Incr. (%)	Exec. Time (sec)		Redc. (%)	Speed Boost (*x*)	Impr. ?
			Rules6	Rules7 @λ		Rules6	Rules7 @λ		Rules6	Rules7 @λ			
1	Adult(M)L	99	191	189	**1.05**	77.90	77.90	**0.00**	400.45	413.23	**-3.19**	nil	no
2	Anneal(M)	95	40	38	**5.00**	97.79	97.27	**-0.53**	0.52	0.43	**17.31**	1.2	no
3	Arrhythmia(M)L	90	110	104	**5.45**	55.25	60.00	**8.60**	82.43	82.17	**0.32**	1.0	yes
4	Breast-Cancer(C)	99	27	25	**7.41**	94.64	94.78	**0.15**	0.09	0.07	**22.22**	1.3	yes
5	Breast-Cancer(N)	95	73	67	**8.22**	66.07	68.93	**4.33**	0.25	0.21	**16.00**	1.2	yes
6	Car(N)	95	94	91	**3.19**	60.64	64.39	**6.18**	0.97	0.73	**24.74**	1.3	yes
7	Chess(N)L	95	47	61	**-29.79**	94.06	94.75	**0.73**	22.58	21.48	**4.87**	1.1	no
8	Connect-4(N)L	95	401	394	**1.75**	66.38	67.39	**1.52**	191.58	169.45	**11.55**	1.1	yes
9	Cover-Type(M)L	95	448	417	**6.92**	56.22	57.17	**1.69**	157.06	106.60	**32.13**	1.5	yes
10	Credit-Approval(M)	99	72	71	**1.39**	80.44	81.32	**1.09**	1.28	1.14	**10.94**	1.1	yes
11	Depression(M)	99	120	114	**5.00**	66.43	65.95	**-0.72**	1.16	1.05	**9.48**	1.1	no
12	Dermatology(M)	90	35	33	**5.71**	88.29	89.43	**1.29**	0.82	0.65	**20.73**	1.3	yes
13	Ecoli(C)	90	50	44	**12.00**	78.39	80.65	**2.88**	0.10	0.08	**20.00**	1.3	yes
14	Flags(M)	99	57	56	**1.75**	58.13	60.00	**3.22**	0.57	0.46	**19.30**	1.2	yes
15	German-Credit(M)	95	239	216	**9.62**	71.90	73.00	**1.53**	7.13	6.16	**13.60**	1.2	yes
16	Hayes-Roth(N)	90	30	18	**40.00**	70.00	72.67	**3.81**	0.02	0.01	**50.00**	2.0	yes
17	Heart-Cleveland(M)	90	97	92	**5.15**	55.36	55.71	**0.63**	0.55	0.49	**10.91**	1.1	yes
18	Heart-Hungarian(M)	99	48	47	**2.08**	76.79	77.14	**0.46**	0.19	0.17	**10.53**	1.1	yes
19	Hepatitis(M)	90	30	29	**3.33**	82.00	82.67	**0.82**	0.11	0.11	**0.00**	nil	yes
20	Horse-Colic(M)	99	69	66	**4.35**	80.83	80.28	**-0.68**	0.64	0.57	**10.94**	1.1	no
21	Hyperthyroid(M)L	99	72	68	**5.56**	97.99	98.10	**0.11**	25.47	21.45	**15.78**	1.2	yes
22	Hypothyroid(M)L	90	65	84	**-29.23**	90.79	95.89	**5.62**	13.00	13.00	**0.00**	nil	no
23	Image(C)	90	38	35	**7.89**	76.67	82.38	**7.45**	0.17	0.15	**11.76**	1.1	yes
24	Ionosphere(C)	95	49	44	**10.20**	87.06	88.24	**1.36**	0.53	0.51	**3.77**	1.0	yes
25	Iris(C)	99	11	10	**9.09**	95.33	93.33	**-2.10**	0.00	0.00	**0.00**	nil	no
26	Lymphography(N)	99	30	29	**3.33**	82.14	83.57	**1.74**	0.11	0.11	**0.00**	nil	yes
27	Mushroom(N)L	95	26	18	**30.77**	97.58	99.70	**2.17**	10.05	5.22	**48.06**	1.9	yes
28	Nursery(N)L	90	358	350	**2.23**	67.99	71.14	**4.63**	118.36	82.28	**30.48**	1.4	yes
29	Parkinsons(C)	95	30	29	**3.33**	90.00	92.78	**3.09**	0.13	0.12	**7.69**	1.1	yes
30	Pendigits(C)L	90	500	460	**8.00**	93.17	94.55	**1.48**	248.15	170.07	**31.46**	1.5	yes
31	Pima-Indians(C)	99	120	100	**16.67**	69.08	71.32	**3.24**	0.96	0.67	**30.21**	1.4	yes
32	P-O-Patient(M)	90	25	23	**8.00**	65.00	70.00	**7.69**	0.03	0.02	**33.33**	1.5	yes
33	Promoters(N)	99	20	18	**10.00**	78.00	86.00	**10.26**	0.15	0.17	**-13.33**	nil	no
34	Soybean-Large(N)	90	55	49	**10.91**	90.00	91.41	**1.57**	1.97	1.60	**18.78**	1.2	yes
35	Spect(C)	99	54	48	**11.11**	78.46	78.46	**0.00**	1.04	0.93	**10.58**	1.1	yes
36	SPECT-Heart(N)	99	34	32	**5.88**	82.69	83.08	**0.47**	0.51	0.45	**11.76**	1.1	yes
37	Splice(N)L	95	254	237	**6.69**	90.32	91.22	**1.00**	173.57	222.21	**-28.02**	nil	no
38	Tic-Tac-Toe(N)	95	28	27	**3.57**	92.74	94.42	**1.81**	0.26	0.22	**15.38**	1.2	yes
39	Vehicle(C)	95	198	185	**6.57**	67.20	67.68	**0.71**	4.38	3.88	**11.42**	1.1	yes
40	Wine(C)	99	29	27	**6.90**	87.50	88.75	**1.43**	0.05	0.04	**20.00**	1.3	yes
Average:			107	101	**5.36**	78.93	80.59	**2.10**	36.68	33.21	**9.47**		

Table 3.9 RULES-7@*λ* vs. RULES-6

No.	Criterion	No. of Datasets for which					
		Redc.		Equal		Incr.	
		Total	%age	Total	%age	Total	%age
1	No. of Rules	38	95%	0	0%	2	5%
2	Accuracy	4	10%	2	5%	34	85%
3	Exec. Time	33	83%	4	10%	3	7%
4	Overall Impr.	31					78%

Table 3.10 Summary of RULES-7@λ

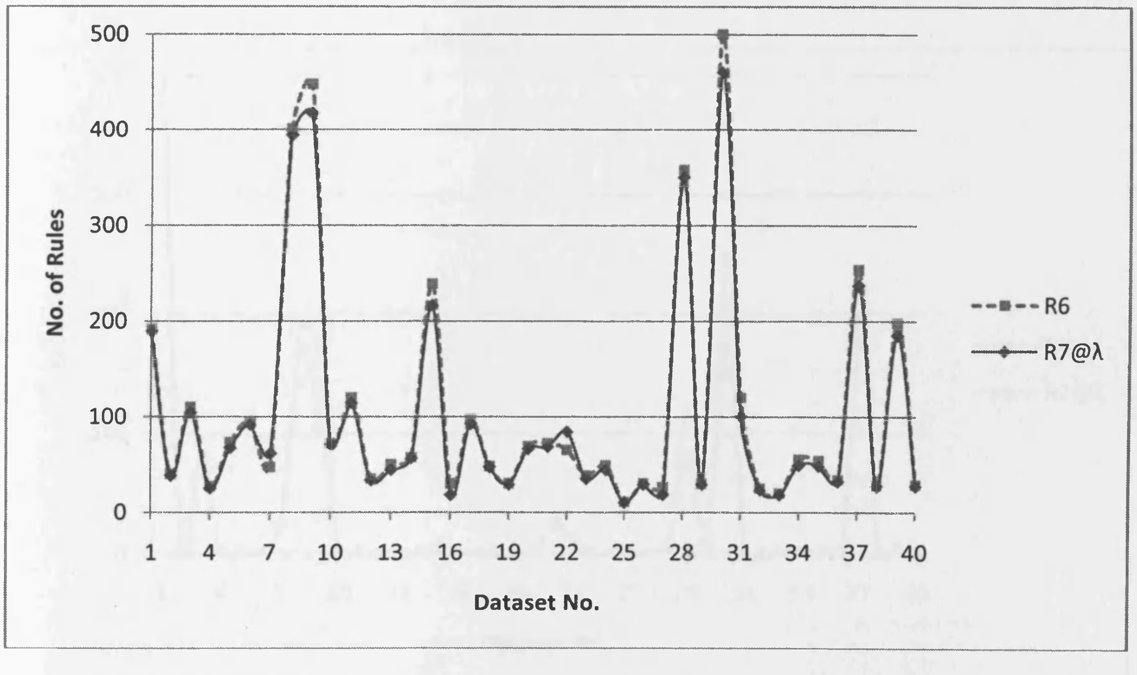


Figure 3.10 RULES-7@λ vs. RULES-6 in terms of Number of Rules

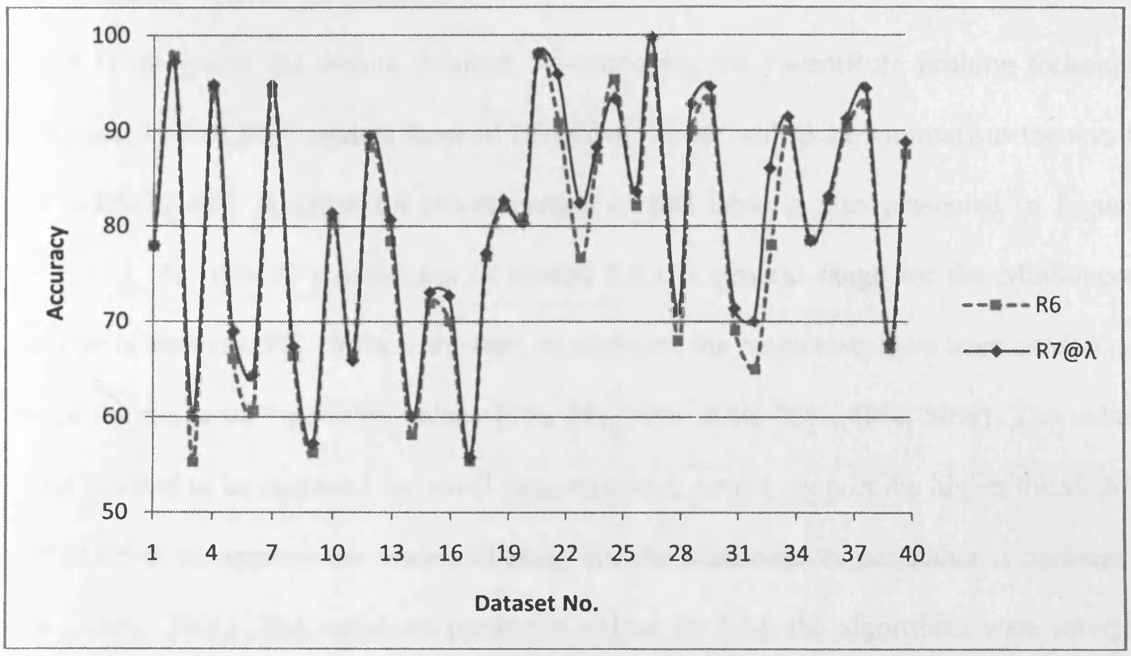


Figure 3.11 RULES-7@λ vs. RULES-6 in terms of Accuracy

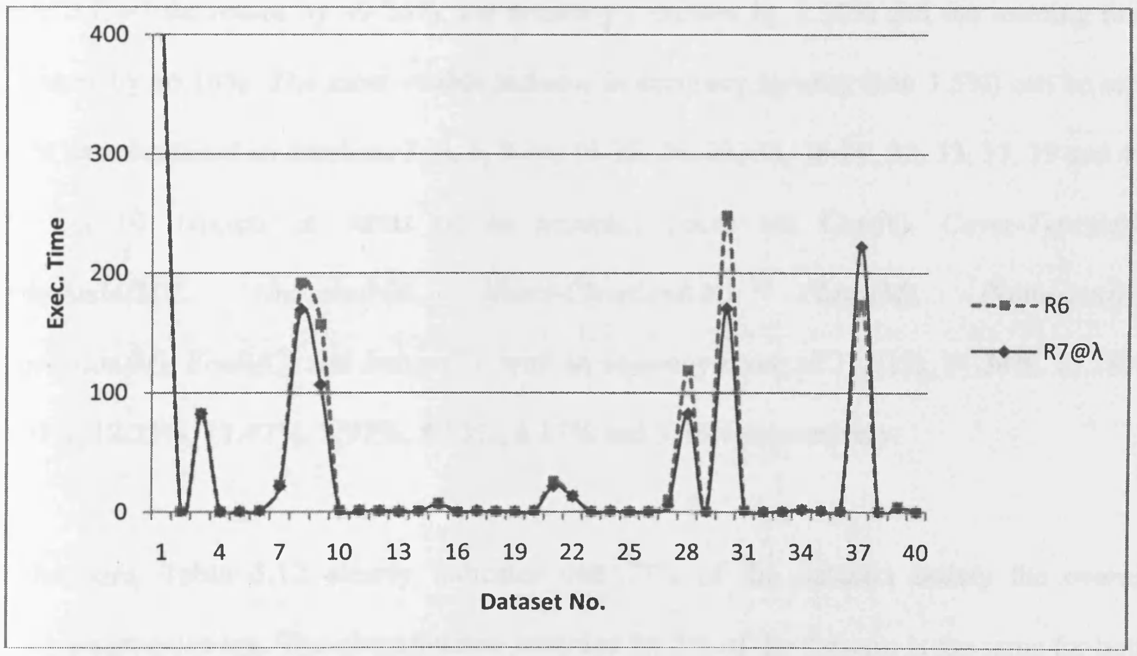


Figure 3.12 RULES-7@λ vs. RULES-6 in terms of Exec. Time

3.5.3 RULES-7@MS vs. RULES-6

Table 3.11 compares the results obtained by employing the ParentRule pruning technique MinSupport in RULES-7 against those of RULES-6, while Table 3.12 summarises the results for RULES-7@MS. A graphical representation of this table is also presented in Figures 3.13 – 3.15. As already pointed out in section 3.4.2, a general range for the MinSupport parameter is between 5% - 95%. However, to minimise the range tests have been carried out only for the range of 7 possible values {1%, 5%, 10%, 20%, 30%, 40%, 50%}. The values 1% and 5% had to be included for small datasets which cannot support the higher thresholds. For RULES-6, an appropriate empirical range for the MinPositives parameter is between 1 and 5 (Afify, 2004). The optimum parameter values for both the algorithms were selected from the point of view of maximum accuracy achieved. On the basis of frequency of occurrence in the tested datasets, 50% may be selected as the default value of MinSupport for RULES-7. It can be seen from the last row in Table 3.11 that the cumulative number of rules for RULES-7 decreases by 49.26%, the accuracy increases by 3.56% and the learning time decreases by 66.16%. The most visible increase in accuracy (greater than 1.5%) can be seen for 24 datasets listed as numbers 3, 5, 6, 9-14, 16-18, 20, 22, 23, 26-29, 32, 33, 37, 39 and 40. The top 10 datasets in terms of an accuracy boost are *Car(N), Cover-Type(M)L, Arrhythmia(M)L, Nursery(N)L, Heart-Cleveland(M), Flags(M), Promoters(N), Depression(M), Ecoli(C)* and *Image(C),* with an accuracy boost of 21.91%, 14.54%, 13.78%, 13.03%, 12.25%, 11.47%, 8.97%, 8.13%, 6.17% and 5.58% respectively.

Furthermore, Table 3.12 clearly indicates that 75% of the datasets satisfy the *overall improvement criterion.* The classification accuracy for 5% of the datasets is the same for both the algorithms. For 75% of the datasets, the accuracy achieved by RULES-7 is higher, whereas for the remaining 20% it is lower. The number of rules induced by RULES-7 is

.	Dataset	Min Pos ∈ {2, 3, 4, 5}	MS ∈ {1, 5, 10, 20, 30, 40, 50} %	No. of Rules		Redc. (%)	Accuracy (%)		Incr. (%)	Exec. Time		Redc. (%)	Speed Boost (*x*)	Impr. ?
				Rules6	Rules7 @MS		Rules6	Rules7 @MS		Rules6	Rules7 @MS			
	Adult(M)L	2	50	191	48	**74.87**	77.90	76.67	**-1.58**	400.45	30.42	**92.40**	**13.2**	no
	Anneal(M)	5	5	40	39	**2.50**	97.92	97.79	**-0.13**	0.52	0.40	**23.08**	**1.3**	no
	Arrhythmia(M)L	4	5	103	115	**-11.65**	56.25	64.00	**13.78**	60.07	72.38	**-20.49**	**nil**	no
	Breast-Cancer(C)	2	1	27	24	**11.11**	94.64	94.06	**-0.61**	0.09	0.08	**11.11**	**1.1**	no
	Breast-Cancer(N)	4	50	71	16	**77.46**	68.21	71.07	**4.19**	0.24	0.02	**91.67**	**12.0**	yes
	Car(N)	5	20	92	6	**93.48**	61.40	74.85	**21.91**	0.90	0.03	**96.67**	**30.0**	yes
	Chess(N)L	2	20	47	29	**38.30**	94.06	94.81	**0.80**	22.58	7.59	**66.39**	**3.0**	yes
	Connect-4(N)L	2	5	401	171	**57.36**	66.38	66.44	**0.09**	191.58	42.69	**77.72**	**4.5**	yes
	Cover-Type(M)L	5	50	446	37	**91.70**	56.68	64.92	**14.54**	151.48	8.07	**94.67**	**18.8**	yes
0	Credit-Approval(M)	4	40	67	28	**58.21**	80.74	84.12	**4.19**	1.15	0.23	**80.00**	**5.0**	yes
1	Depression(M)	3	50	119	23	**80.67**	67.38	72.86	**8.13**	1.15	0.10	**91.30**	**11.5**	yes
2	Dermatology(M)	2	20	35	30	**14.29**	88.29	92.00	**4.20**	0.82	0.33	**59.76**	**2.5**	yes
3	Ecoli(C)	2	10	50	46	**8.00**	78.39	83.23	**6.17**	0.10	0.07	**30.00**	**1.4**	yes
4	Flags(M)	4	30	57	22	**61.40**	60.00	66.88	**11.47**	0.45	0.07	**84.44**	**6.4**	yes
5	German-Credit(M)	2	20	239	100	**58.16**	71.90	72.30	**0.56**	7.13	1.35	**81.07**	**5.3**	yes
6	Hayes-Roth(N)	2	5	30	18	**40.00**	70.00	72.00	**2.86**	0.02	0.01	**50.00**	**2.0**	yes
7	Heart-Cleveland(M)	2	50	97	18	**81.44**	55.36	62.14	**12.25**	0.55	0.03	**94.55**	**18.3**	yes
8	Heart-Hungarian(M)	5	30	46	28	**39.13**	77.86	81.43	**4.59**	0.18	0.05	**72.22**	**3.6**	yes
9	Hepatitis(M)	5	20	30	24	**20.00**	82.67	82.67	**0.00**	0.11	0.07	**36.36**	**1.6**	yes
0	Horse-Colic(M)	3	50	70	26	**62.86**	81.11	83.06	**2.40**	0.62	0.05	**91.94**	**12.4**	yes
1	Hyperthyroid(M)L	2	1	72	67	**6.94**	97.99	97.89	**-0.10**	25.47	19.32	**24.15**	**1.3**	no
2	Hypothyroid(M)L	2	50	65	22	**66.15**	90.79	95.57	**5.26**	13.00	1.88	**85.54**	**6.9**	yes
3	Image(C)	2	5	37	37	**0.00**	76.67	80.95	**5.58**	0.17	0.14	**17.65**	**1.2**	yes
4	Ionosphere(C)	4	40	49	28	**42.86**	88.24	89.12	**1.00**	0.53	0.09	**83.02**	**5.9**	yes
5	Iris(C)	2	1	11	10	**9.09**	95.33	93.33	**-2.10**	0.00	0.00	**0.00**	**nil**	no
6	Lymphography(N)	3	20	32	30	**6.25**	82.86	85.00	**2.58**	0.11	0.08	**27.27**	**1.4**	yes
7	Mushroom(N)L	2	10	26	24	**7.69**	97.58	99.56	**2.03**	10.05	6.54	**34.93**	**1.5**	yes
8	Nursery(N)L	2	10	358	50	**86.03**	67.99	76.85	**13.03**	118.36	6.73	**94.31**	**17.6**	yes
9	Parkinsons(C)	5	1	30	29	**3.33**	90.56	93.33	**3.06**	0.13	0.13	**0.00**	**nil**	yes
0	Pendigits(C)L	4	1	500	456	**8.80**	93.19	93.85	**0.71**	242.60	234.91	**3.17**	**1.0**	yes
1	Pima-Indians(C)	3	5	117	68	**41.88**	69.34	70.13	**1.14**	0.94	0.36	**61.70**	**2.6**	yes
2	P-O-Patient(M)	2	50	25	12	**52.00**	65.00	67.50	**3.85**	0.03	0.01	**66.67**	**3.0**	yes
3	Promoters(N)	2	50	20	14	**30.00**	78.00	85.00	**8.97**	0.15	0.05	**66.67**	**3.0**	yes
4	Soybean-Large(N)	5	5	53	51	**3.77**	90.63	90.47	**-0.18**	1.75	1.42	**18.86**	**1.2**	no
5	Spect(C)	5	1	54	53	**1.85**	78.85	78.46	**-0.49**	1.01	1.04	**-2.97**	**nil**	no
6	SPECT-Heart(N)	2	1	34	34	**0.00**	82.69	82.69	**0.00**	0.51	0.45	**11.76**	**1.1**	no
7	Splice(N)L	4	10	254	128	**49.61**	90.35	93.73	**3.74**	167.24	43.45	**74.02**	**3.8**	yes
8	Tic-Tac-Toe(N)	2	1	28	28	**0.00**	92.74	92.63	**-0.12**	0.26	0.24	**7.69**	**1.1**	no
9	Vehicle(C)	5	5	189	134	**29.10**	68.05	69.15	**1.62**	4.09	1.87	**54.28**	**2.2**	yes
0	Wine(C)	2	10	29	29	**0.00**	87.50	91.88	**5.01**	0.05	0.04	**20.00**	**1.3**	yes
	Average:			106	54	**49.26**	79.29	82.11	**3.56**	35.67	12.07	**66.16**		

Table 3.11 RULES-7@MS vs. RULES-6

No.	Criterion	No. of Datasets for which					
		Redc.		Equal		Incr.	
		Total	%age	Total	%age	Total	%age
1	No. of Rules	35	88%	4	10%	1	2%
2	Accuracy	8	20%	2	5%	30	75%
3	Exec. Time	36	90%	2	5%	2	5%
4	Overall Impr.			30			75%

Table 3.12 Summary of RULES-7@MS

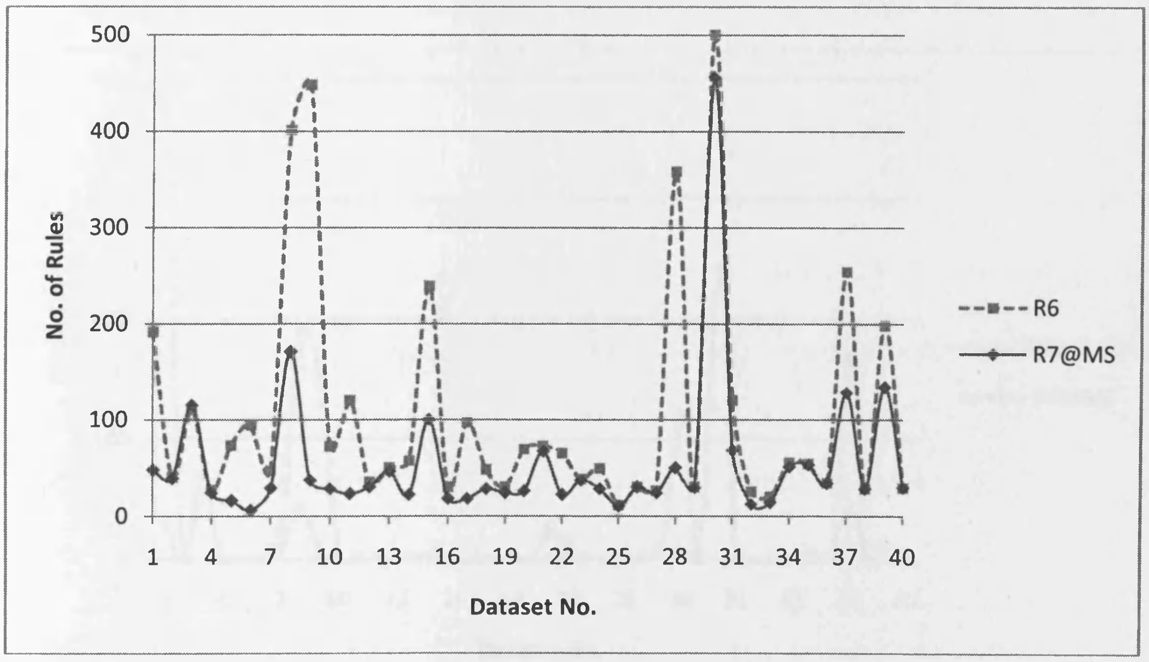


Figure 3.13 RULES-7@MS vs. RULES-6 in terms of Number of Rules

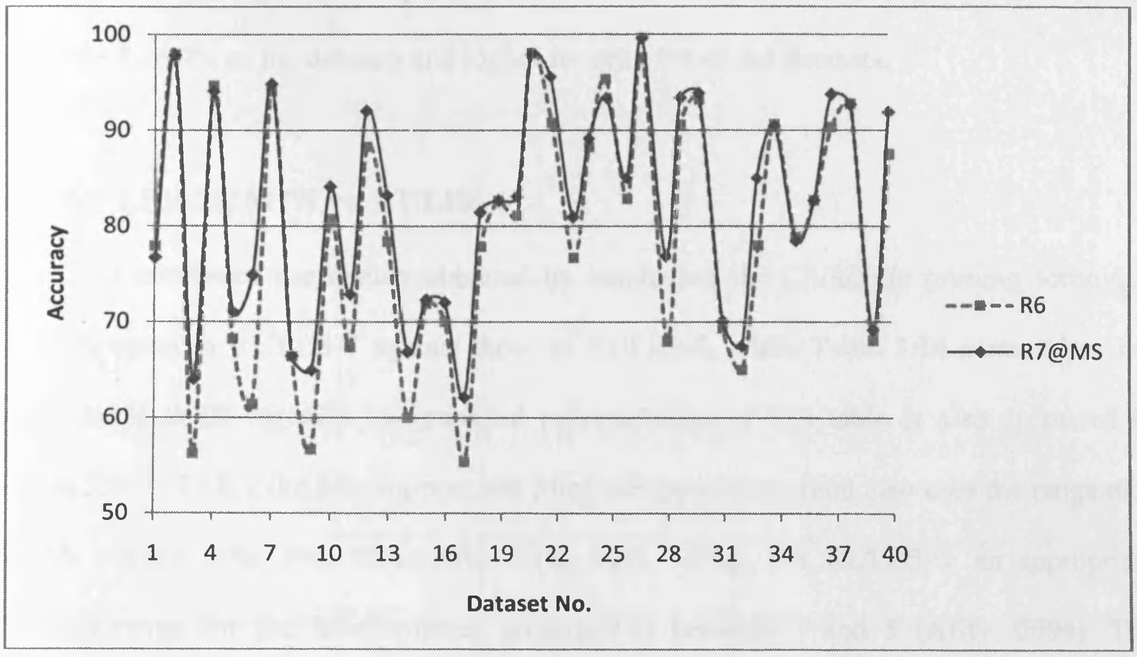


Figure 3.14 RULES-7@MS vs. RULES-6 in terms of Accuracy

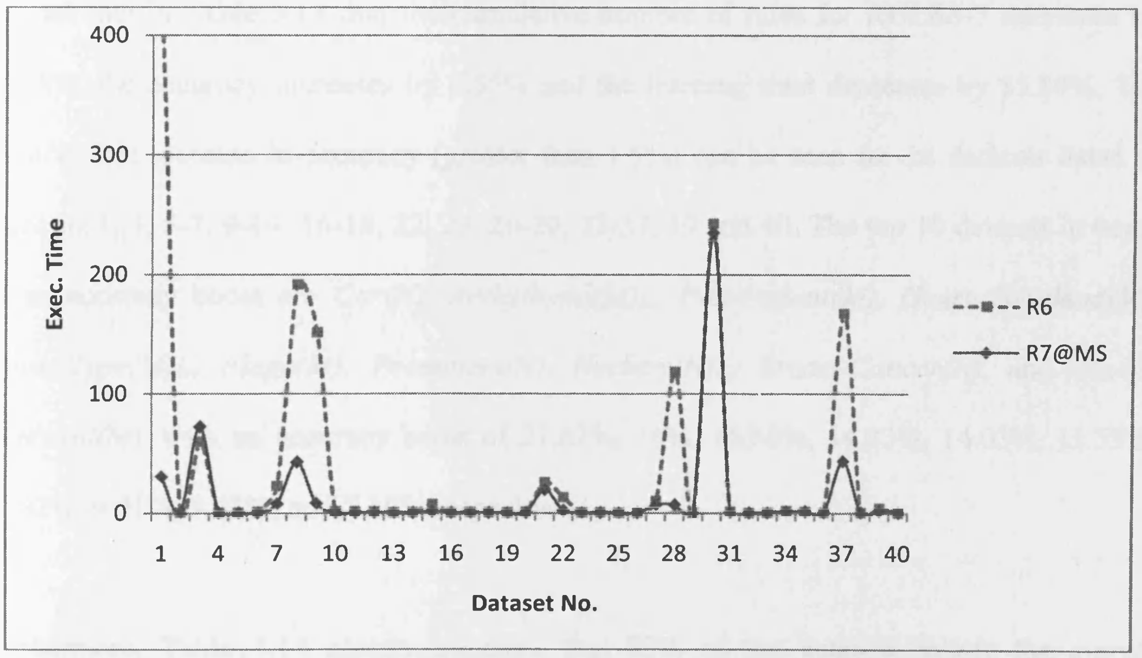


Figure 3.15 RULES-7@MS vs. RULES-6 in terms of Exec. Time

lower for 88% and higher for only 2% of the datasets, whereas the learning time for RULES-7 is lower for 90% of the datasets and higher for only 5% of the datasets.

3.5.4 RULES-7@MPS vs. RULES-6

Table 3.13 compares the results obtained by employing the *ChildRule* pruning technique MinPosSupport in RULES-7 against those of RULES-6, while Table 3.14 summarises the results for RULES-7@MPS. A graphical representation of this table is also presented in Figures 3.16 – 3.18. Like MinSupport, the MinPosSupport constraint also uses the range of 7 possible values {1%, 5%, 10%, 20%, 30%, 40%, 50%}. For RULES-6, an appropriate empirical range for the MinPositives parameter is between 1 and 5 (Afify, 2004). The optimum parameter values for both the algorithms were selected from the point of view of maximum accuracy achieved. On the basis of frequency of occurrence in the tested datasets, 5% may be selected as the default value of MinPosSupport for RULES-7. It can be seen from the last row in Table 3.13 that the cumulative number of rules for RULES-7 decreases by 40.08%, the accuracy increases by 4.55% and the learning time decreases by 55.89%. The most visible increase in accuracy (greater than 1.5%) can be seen for 28 datasets listed as numbers 1, 3, 5-7, 9-14, 16-18, 22, 23, 26-29, 32-37, 39 and 40. The top 10 datasets in terms of an accuracy boost are *Car(N), Arrhythmia(M)L, P-O-Patient(M), Heart-Cleveland(M), Cover-Type(M)L, Flags(M), Promoters(N), Nursery(N)L, Breast-Cancer(N),* and *Credit-Approval(M),* with an accuracy boost of 27.62%, 16%, 15.38%, 14.83%, 14.03%, 13.55%, 12.82%, 9.91%, 9.43%, and 7.10% respectively.

Furthermore, Table 3.14 clearly indicates that 83% of the datasets satisfy the *overall improvement criterion*. The classification accuracy for 7% of the datasets is the same for both the algorithms. For 83% of the datasets, the accuracy achieved by RULES-7 is higher,

o.	Dataset	Min Pos ∈ {2, 3, 4, 5}	MPS ∈ {1, 5, 10, 20, 30, 40, 50} %	No. of Rules		Redc. (%)	Accuracy (%)		Incr. (%)	Exec. Time		Redc. (%)	Speed Boost (x)	Impr. ?
				Rules6	Rules7 @MPS		Rules6	Rules7 @MPS		Rules6	Rules7 @MPS			
1	Adult(M)L	2	5	191	86	**54.97**	77.90	79.90	**2.57**	400.45	97.46	**75.66**	**4.1**	yes
2	Anneal(M)	5	1	40	44	**-10.00**	97.92	97.40	**-0.53**	0.52	0.52	**0.00**	**nil**	no
3	Arrhythmia(M)L	4	50	103	52	**49.51**	56.25	65.25	**16.00**	60.07	42.75	**28.83**	**1.4**	yes
4	Breast-Cancer(C)	2	20	27	13	**51.85**	94.64	95.36	**0.76**	0.09	0.04	**55.56**	**2.3**	yes
5	Breast-Cancer(N)	4	40	71	10	**85.92**	68.21	74.64	**9.43**	0.24	0.01	**95.83**	**24.0**	yes
6	Car(N)	5	20	92	6	**93.48**	61.40	78.36	**27.62**	0.90	0.03	**96.67**	**30.0**	yes
7	Chess(N)L	2	5	47	48	**-2.13**	94.06	95.79	**1.84**	22.58	17.92	**20.64**	**1.3**	no
8	Connect-4(N)L	2	1	401	363	**9.48**	66.38	66.11	**-0.41**	191.58	118.25	**38.28**	**1.6**	no
9	Cover-Type(M)L	5	5	446	179	**59.87**	56.68	64.63	**14.03**	151.48	62.08	**59.02**	**2.4**	yes
0	Credit-Approval(M)	4	40	67	19	**71.64**	80.74	86.47	**7.10**	1.15	0.16	**86.09**	**7.2**	yes
1	Depression(M)	3	30	119	16	**86.55**	67.38	72.14	**7.06**	1.15	0.08	**93.04**	**14.4**	yes
2	Dermatology(M)	2	40	35	33	**5.71**	88.29	92.00	**4.20**	0.82	0.67	**18.29**	**1.2**	yes
3	Ecoli(C)	2	5	50	49	**2.00**	78.39	80.32	**2.46**	0.10	0.08	**20.00**	**1.3**	yes
4	Flags(M)	4	50	57	24	**57.89**	60.00	68.13	**13.55**	0.45	0.21	**53.33**	**2.1**	yes
5	German-Credit(M)	2	5	239	143	**40.17**	71.90	72.20	**0.42**	7.13	3.63	**49.09**	**2.0**	yes
6	Hayes-Roth(N)	2	5	30	18	**40.00**	70.00	72.00	**2.86**	0.02	0.01	**50.00**	**2.0**	yes
7	Heart-Cleveland(M)	2	40	97	23	**76.29**	55.36	63.57	**14.83**	0.55	0.05	**90.91**	**11.0**	yes
8	Heart-Hungarian(M)	5	50	46	11	**76.09**	77.86	82.86	**6.42**	0.18	0.01	**94.44**	**18.0**	yes
9	Hepatitis(M)	5	20	30	26	**13.33**	82.67	82.67	**0.00**	0.11	0.08	**27.27**	**1.4**	yes
0	Horse-Colic(M)	3	1	70	69	**1.43**	81.11	79.72	**-1.71**	0.62	0.59	**4.84**	**1.1**	no
1	Hyperthyroid(M)L	2	1	72	64	**11.11**	97.99	97.94	**-0.05**	25.47	19.45	**23.64**	**1.3**	no
2	Hypothyroid(M)L	2	5	65	47	**27.69**	90.79	96.11	**5.86**	13.00	8.92	**31.38**	**1.5**	yes
3	Image(C)	2	20	37	32	**13.51**	76.67	80.95	**5.58**	0.17	0.12	**29.41**	**1.4**	yes
4	Ionosphere(C)	4	5	49	49	**0.00**	88.24	88.24	**0.00**	0.53	0.53	**0.00**	**nil**	no
5	Iris(C)	2	1	11	11	**0.00**	95.33	96.67	**1.41**	0.00	0.00	**0.00**	**nil**	no
6	Lymphography(N)	3	20	32	26	**18.75**	82.86	86.43	**4.31**	0.11	0.08	**27.27**	**1.4**	yes
7	Mushroom(N)L	2	5	26	24	**7.69**	97.58	99.65	**2.12**	10.05	7.29	**27.46**	**1.4**	yes
8	Nursery(N)L	2	5	358	54	**84.92**	67.99	74.73	**9.91**	118.36	10.83	**90.85**	**10.9**	yes
9	Parkinsons(C)	5	1	30	29	**3.33**	90.56	93.33	**3.06**	0.13	0.13	**0.00**	**nil**	yes
0	Pendigits(C)L	4	1	500	487	**2.60**	93.19	94.47	**1.37**	242.60	171.05	**29.49**	**1.4**	yes
1	Pima-Indians(C)	3	1	117	75	**35.90**	69.34	69.61	**0.39**	0.94	0.49	**47.87**	**1.9**	yes
2	P-O-Patient(M)	2	50	25	5	**80.00**	65.00	75.00	**15.38**	0.03	0.00	**100.00**	**max**	yes
3	Promoters(N)	2	30	20	16	**20.00**	78.00	88.00	**12.82**	0.15	0.06	**60.00**	**2.5**	yes
4	Soybean-Large(N)	5	10	53	52	**1.89**	90.63	92.03	**1.54**	1.75	1.60	**8.57**	**1.1**	yes
5	Spect(C)	5	30	54	24	**55.56**	78.85	82.31	**4.39**	1.01	0.29	**71.29**	**3.5**	yes
6	SPECT-Heart(N)	2	40	34	22	**35.29**	82.69	85.00	**2.79**	0.51	0.27	**47.06**	**1.9**	yes
7	Splice(N)L	4	10	254	110	**56.69**	90.35	92.48	**2.36**	167.24	60.95	**63.56**	**2.7**	yes
8	Tic-Tac-Toe(N)	2	1	28	27	**3.57**	92.74	92.74	**0.00**	0.26	0.22	**15.38**	**1.2**	yes
9	Vehicle(C)	5	5	189	127	**32.80**	68.05	70.00	**2.87**	4.09	2.38	**41.81**	**1.7**	yes
0	Wine(C)	2	1	29	28	**3.45**	87.50	90.63	**3.58**	0.05	0.05	**0.00**	**nil**	yes
Average:				106	64	**40.08**	79.29	82.90	**4.55**	35.67	15.73	**55.89**		

Table 3.13 RULES-7@MPS vs. RULES-6

No.	Criterion	No. of Datasets for which					
		Redc.		Equal		Incr.	
		Total	%age	Total	%age	Total	%age
1	No. of Rules	36	90%	2	5%	2	5%
2	Accuracy	4	10%	3	7%	33	83%
3	Exec. Time	35	88%	5	12%	0	0%
4	Overall Impr.	33					83%

Table 3.14 Summary of RULES-7@MPS

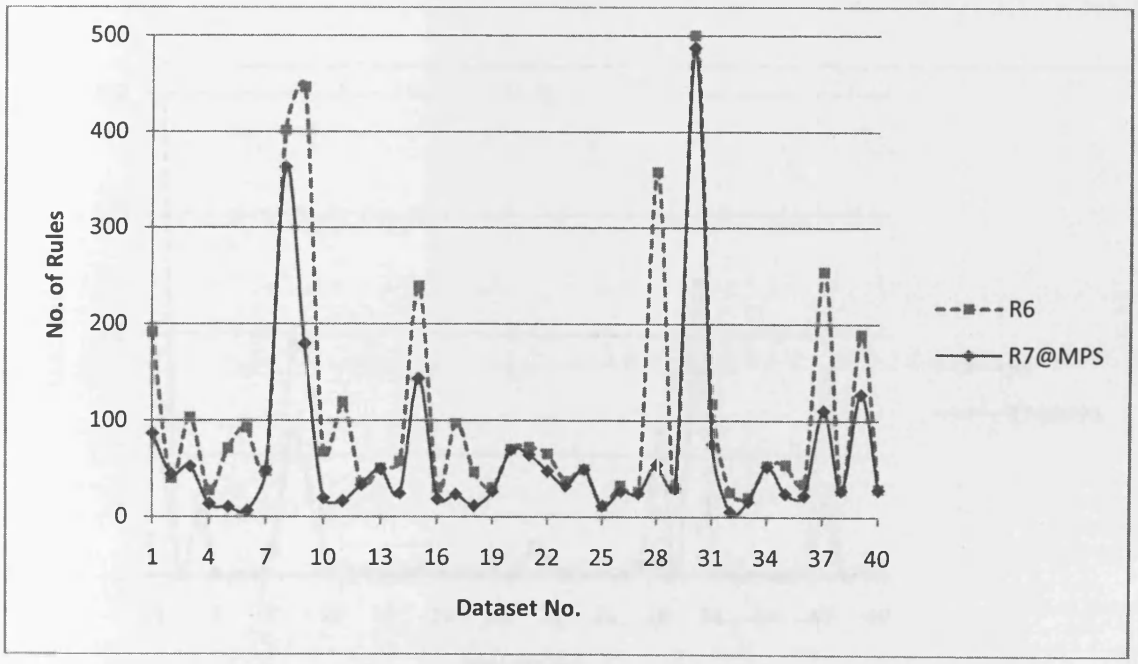


Figure 3.16 RULES-7@MPS vs. RULES-6 in terms of Number of Rules

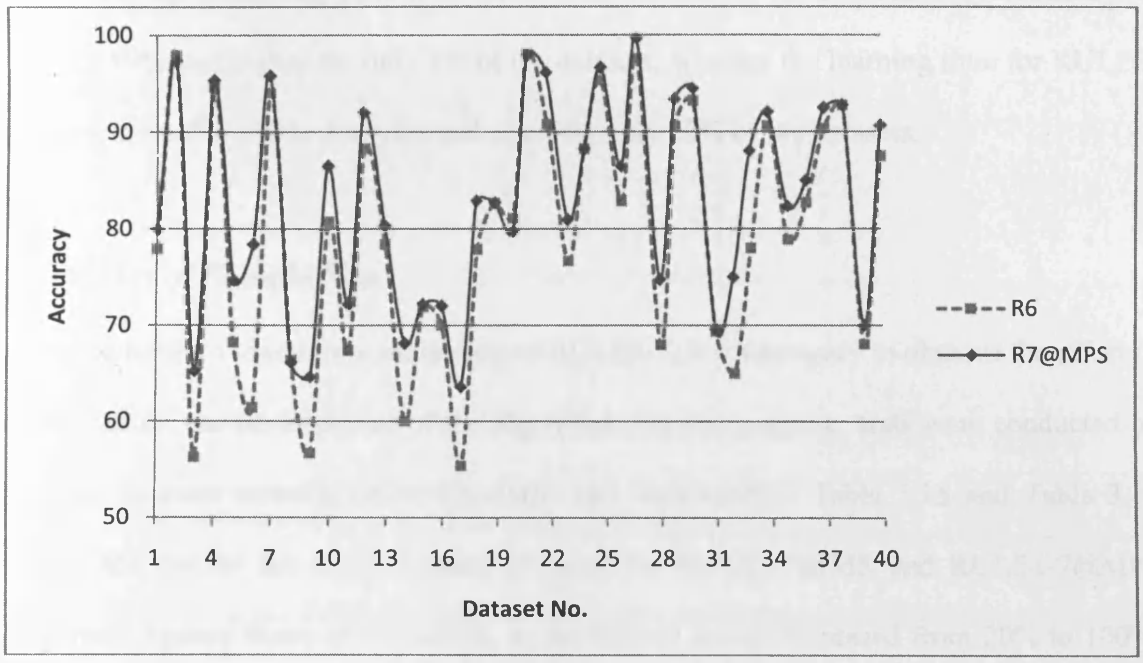


Figure 3.17 RULES-7@MPS vs. RULES-6 in terms of Accuracy

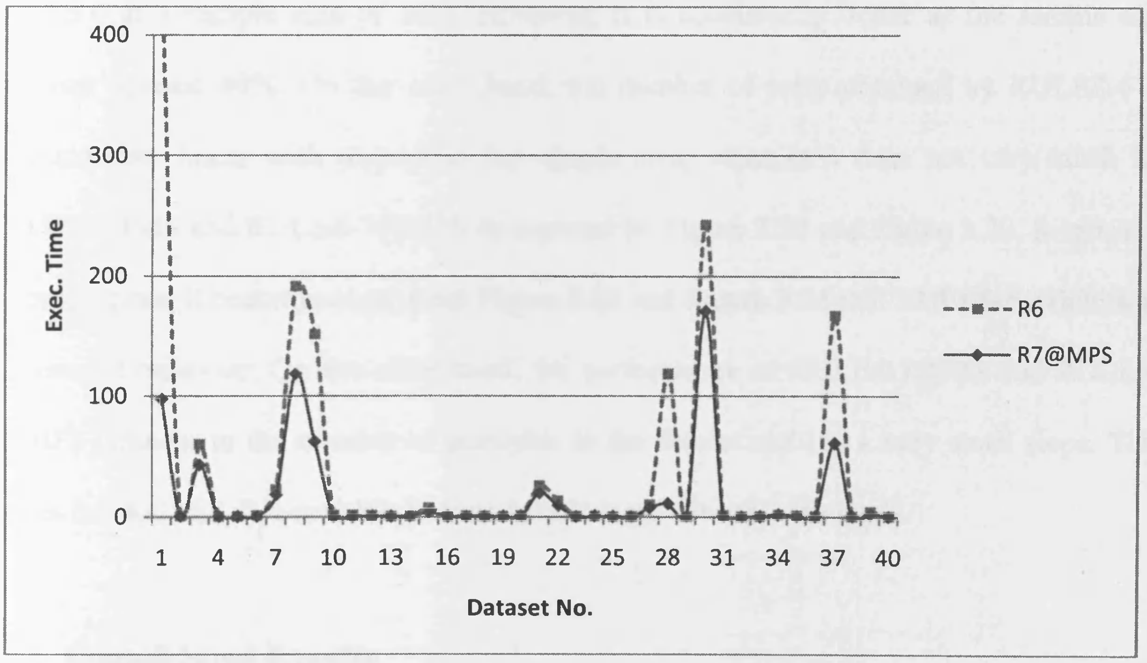


Figure 3.18 RULES-7@MPS vs. RULES-6 in terms of Exec. Time

whereas for the remaining 10% it is lower. The number of rules induced by RULES-7 is lower for 90% and higher for only 5% of the datasets, whereas the learning time for RULES-7 is lower for 88%, higher for none and equal for only 12% of the datasets.

3.5.5 Effect of Sample Size

In order to have an idea of the scalability of RULES-7, it is necessary to observe the effect of sample size on the performance of the algorithm. For this purpose, tests were conducted on two large datasets namely *Cover-Type(M)L* and *Nursery(N)L*. Table 3.15 and Table 3.16 compare the results for these datasets obtained by RULES-7@MS and RULES-7@MPS respectively against those of RULES-6, as the sample size is increased from 20% to 100%. Furthermore, Figures 3.19 – 3.24 serve to illustrate the effect of sample size on these two pruning techniques in a graphical form. It can be seen from Figure 3.21 and Figure 3.22 that the classification accuracy for these two pruning techniques is a bit less as compared to RULES-6 at a sample size of 20%. However, it is consistently better as the sample size increases beyond 40%. On the other hand, the number of rules obtained by RULES-6 is approximately linear with respect to the sample size, whereas it does not vary much for RULES-7@MS and RULES-7@MPS as depicted by Figure 3.19 and Figure 3.20. In terms of execution time, it becomes clear from Figure 3.23 and Figure 3.24 that RULES-6 exhibits an exponential behavior. On the other hand, the performance of RULES-7@MS and RULES-7@MPS is linear in the number of examples in the dataset and has a very small slope. This proves that RULES-7 is scalable and can handle large datasets efficiently.

3.5.6 Consolidated Results

Table 3.17 consolidates the results presented in sections 3.5.1 – 3.5.4 and highlights the best values by formatting them in bold. A comparison of the efficacy of the new techniques

No.	Dataset	Sample Size	No. of Rules		Redc. (%)	Accuracy (%)		Incr. (%)	Exec. Time (sec)		Redc. (%)
			RULES6	RULES7 @MS		RULES6	RULES7 @MS		RULES6	RULES7 @MS	
1	Cover-Type(M)L	20%	99	30	**69.70**	57.80	50.00	**-13.49**	3.25	0.21	**93.54**
		40%	196	27	**86.22**	51.82	57.60	**11.15**	14.62	0.73	**95.01**
		60%	324	30	**90.74**	49.07	62.25	**26.86**	40.81	2.87	**92.97**
		80%	429	43	**89.98**	54.24	58.82	**8.44**	81.27	5.66	**93.04**
		100%	448	37	**91.74**	56.22	64.92	**15.47**	157.06	8.07	**94.86**
2	Nursery(N)L	20%	33	16	**51.52**	98.68	97.67	**-1.02**	0.69	0.16	**76.81**
		40%	88	23	**73.86**	83.37	87.85	**5.37**	4.43	0.62	**86.00**
		60%	120	27	**77.50**	84.35	86.13	**2.11**	11.71	1.42	**87.87**
		80%	213	39	**81.69**	69.55	75.38	**8.38**	34.25	2.74	**92.00**
		100%	358	50	**86.03**	67.99	76.85	**13.03**	118.36	6.73	**94.31**

Table 3.15 RULES-7@MS vs. RULES-6 with respect to Sample Size

No.	Dataset	Sample Size	No. of Rules		Redc. (%)	Accuracy (%)		Incr. (%)	Exec. Time (sec)		Redc. (%)
			RULES6	RULES7 @MPS		RULES6	RULES7 @MPS		RULES6	RULES7 @MPS	
1	Cover-Type(M)L	20%	99	79	**20.20**	57.80	54.92	**-4.98**	3.25	2.67	**17.85**
		40%	196	125	**36.22**	51.82	51.82	**0.00**	14.62	9.71	**33.58**
		60%	324	142	**56.17**	49.07	55.33	**12.76**	40.81	23.28	**42.96**
		80%	429	167	**61.07**	54.24	61.02	**12.50**	81.27	44.08	**45.76**
		100%	448	179	**60.04**	56.22	64.63	**14.96**	157.06	62.08	**60.47**
2	Nursery(N)L	20%	33	20	**39.39**	98.68	96.73	**-1.98**	0.69	0.32	**53.62**
		40%	88	31	**64.77**	83.37	86.09	**3.26**	4.43	1.14	**74.27**
		60%	120	35	**70.83**	84.35	87.19	**3.37**	11.71	2.27	**80.61**
		80%	213	42	**80.28**	69.55	71.86	**3.32**	34.25	4.58	**86.63**
		100%	358	54	**84.92**	67.99	74.73	**9.91**	118.36	10.83	**90.85**

Table 3.16 RULES-7@MPS vs. RULES-6 with respect to Sample Size

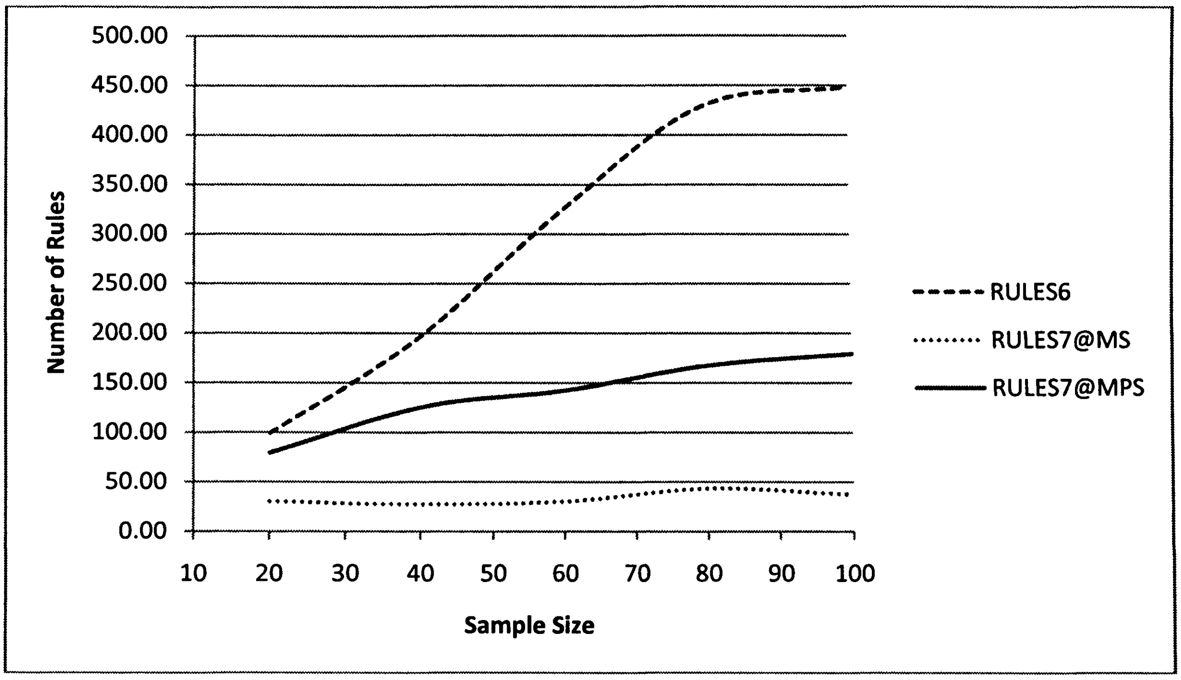


Figure 3.19 Number of Rules vs. Sample Size for *Cover-Type* Data

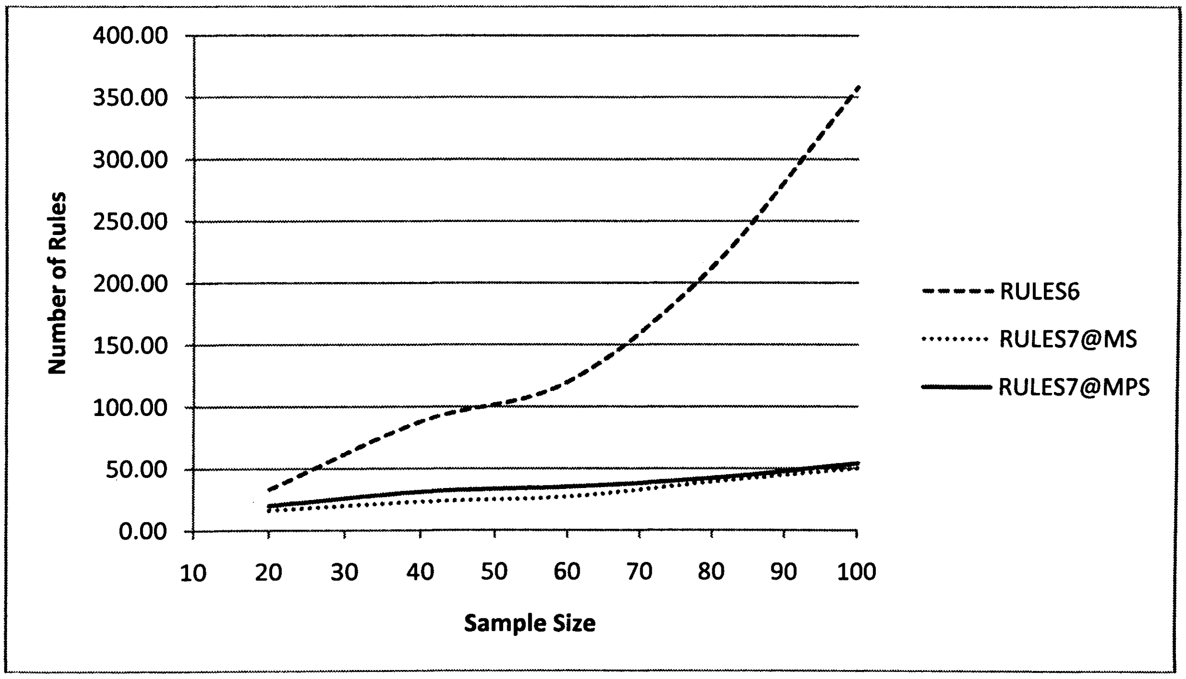


Figure 3.20 Number of Rules vs. Sample Size for *Nursery* Data

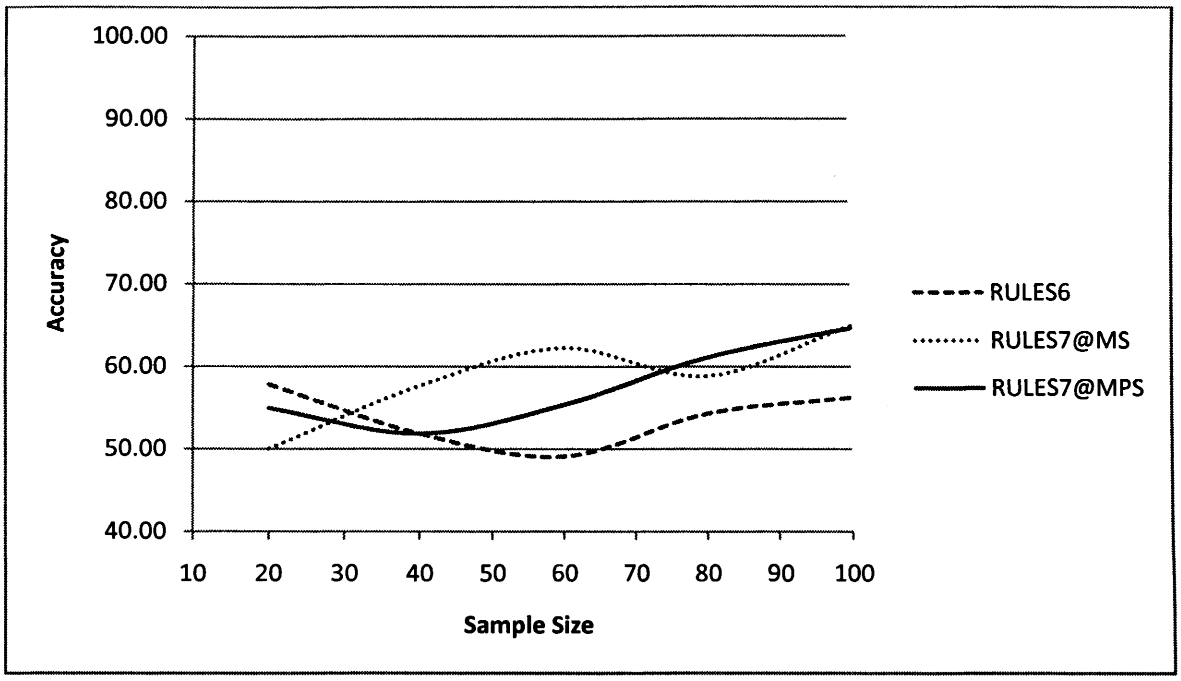


Figure 3.21 Accuracy vs. Sample Size for *Cover-Type* Data

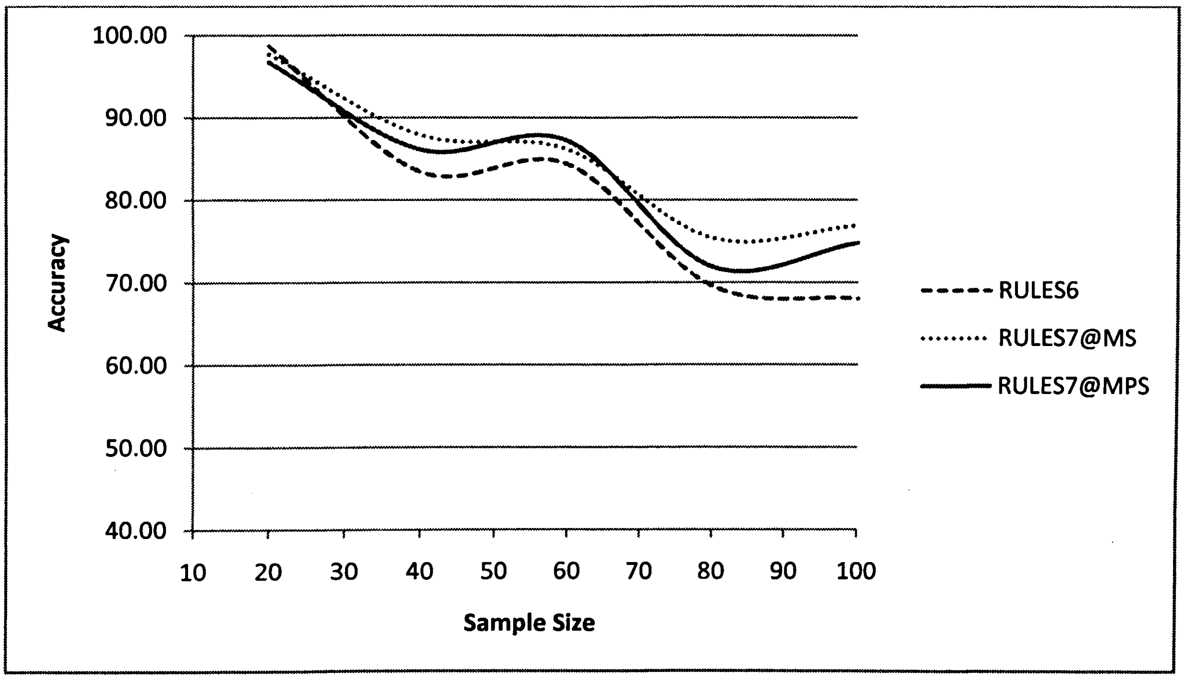


Figure 3.22 Accuracy vs. Sample Size for *Nursery* Data

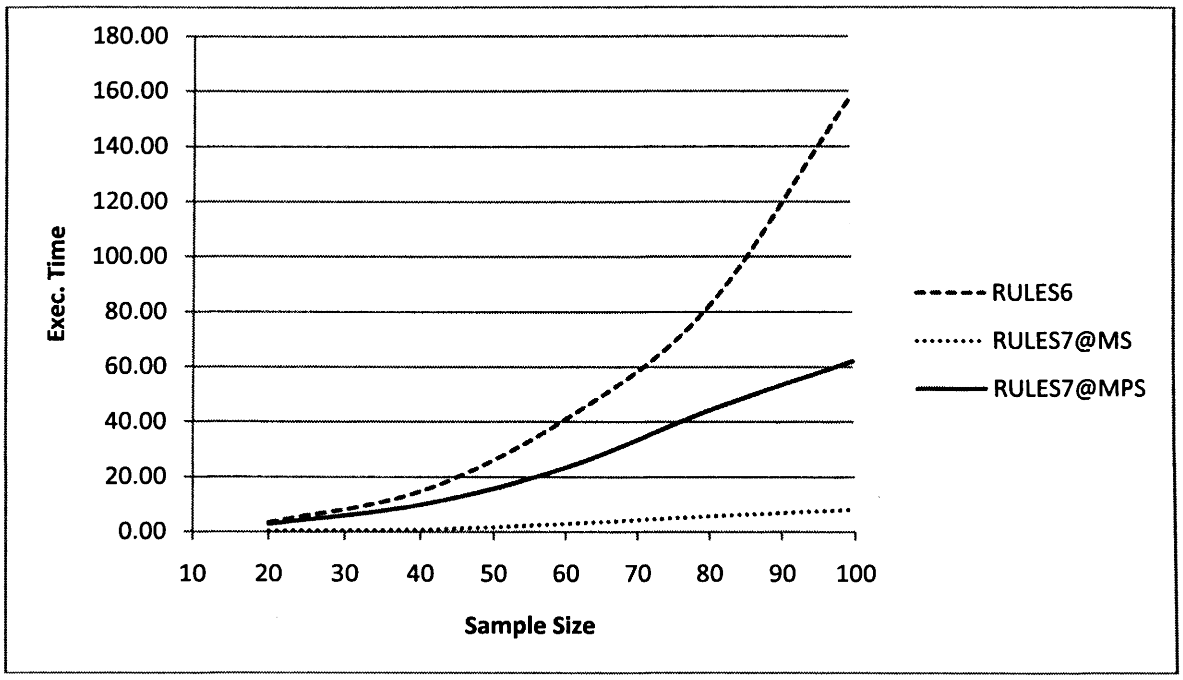


Figure 3.23 Exec. Time vs. Sample Size for *Cover-Type* Data

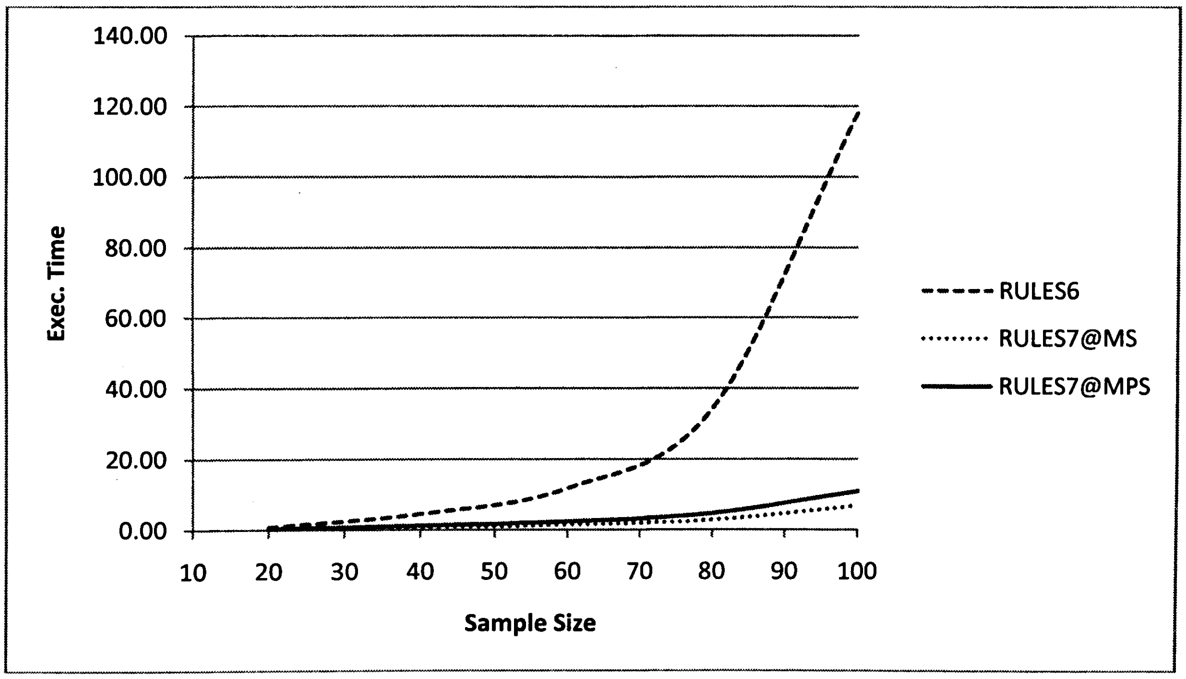


Figure 3.24 Exec. Time vs. Sample Size for *Nursery* Data

No.	Dataset	No. of Rules					Accuracy (%)					Exec. Time (sec)				
		R6	R7	R7 @λ	R7 @MS	R7 @MPS	R6	R7	R7 @λ	R7 @MS	R7 @MPS	R6	R7	R7 @λ	R7 @MS	R7 @MPS
1	Adult(M)L	191	188	189	**48**	86	77.90	77.91	77.90	76.67	**79.90**	400.45	396.67	413.23	**30.42**	97.46
2	Anneal(M)	40	40	**38**	39	44	**97.79**	**97.79**	97.27	**97.79**	97.40	0.52	0.45	0.43	**0.40**	0.52
3	Arrhythmia(M)L	110	110	104	115	**52**	55.25	58.50	60.00	64.00	**65.25**	82.43	87.33	82.17	72.38	**42.75**
4	Breast-Cancer(C)	27	25	25	24	**13**	94.64	94.64	94.78	94.06	**95.36**	0.09	0.09	0.07	0.08	**0.04**
5	Breast-Cancer(N)	73	72	67	16	**10**	66.07	66.79	68.93	71.07	**74.64**	0.25	0.22	0.21	0.02	**0.01**
6	Car(N)	94	94	91	**6**	**6**	60.64	61.70	64.39	74.85	**78.36**	0.97	0.73	0.73	**0.03**	**0.03**
7	Chess(N)L	47	48	61	**29**	48	94.06	94.21	94.75	94.81	**95.79**	22.58	17.37	21.48	**7.59**	17.92
8	Connect-4(N)L	401	400	394	**171**	363	66.38	66.74	**67.39**	66.44	66.11	191.58	129.89	169.45	**42.69**	118.25
9	Cover-Type(M)L	448	449	417	**37**	179	56.22	56.42	57.17	**64.92**	64.63	157.06	127.43	106.60	**8.07**	62.08
10	Credit-Approval(M)	72	72	71	28	**19**	80.44	81.03	81.32	84.12	**86.47**	1.28	1.19	1.14	0.23	**0.16**
11	Depression(M)	120	118	114	23	**16**	66.43	65.48	65.95	**72.86**	72.14	1.16	1.10	1.05	0.10	**0.08**
12	Dermatology(M)	35	36	33	**30**	33	88.29	88.57	89.43	**92.00**	**92.00**	0.82	0.70	0.65	**0.33**	0.67
13	Ecoli(C)	50	48	**44**	46	49	78.39	80.00	80.65	**83.23**	80.32	0.10	0.08	0.08	**0.07**	0.08
14	Flags(M)	57	57	56	**22**	24	58.13	58.75	60.00	66.88	**68.13**	0.57	0.47	0.46	**0.07**	0.21
15	German-Credit(M)	239	239	216	**100**	143	71.90	72.40	**73.00**	72.30	72.20	7.13	6.71	6.16	**1.35**	3.63
16	Hayes-Roth(N)	30	**18**	**18**	**18**	**18**	70.00	72.00	**72.67**	72.00	72.00	0.02	**0.01**	**0.01**	**0.01**	**0.01**
17	Heart-Cleveland(M)	97	97	92	**18**	23	55.36	55.36	55.71	62.14	**63.57**	0.55	0.50	0.49	**0.03**	0.05
18	Heart-Hungarian(M)	48	48	47	28	**11**	76.79	77.50	77.14	81.43	**82.86**	0.19	0.18	0.17	0.05	**0.01**
19	Hepatitis(M)	30	30	29	**24**	26	82.00	80.00	**82.67**	**82.67**	**82.67**	0.11	0.11	0.11	**0.07**	0.08
20	Horse-Colic(M)	69	69	66	**26**	69	80.83	79.72	80.28	**83.06**	79.72	0.64	0.60	0.57	**0.05**	0.59

Table 3.17 Consolidated Results

No.	Dataset	No. of Rules					Accuracy (%)					Exec. Time (sec)				
		R6	R7	R7 @λ	R7 @MS	R7 @MPS	R6	R7	R7 @λ	R7 @MS	R7 @MPS	R6	R7	R7 @λ	R7 @MS	R7 @MPS
21	Hyperthyroid(M)L	72	68	68	67	**64**	97.99	97.99	**98.10**	97.89	97.94	25.47	21.44	21.45	**19.32**	19.45
22	Hypothyroid(M)L	65	65	84	**22**	47	90.79	90.85	95.89	95.57	**96.11**	13.00	11.22	13.00	**1.88**	8.92
23	Image(C)	38	37	35	37	**32**	76.67	80.48	**82.38**	80.95	80.95	0.17	0.17	0.15	0.14	**0.12**
24	Ionosphere(C)	49	48	44	**28**	49	87.06	87.35	88.24	**89.12**	88.24	0.53	0.53	0.51	**0.09**	0.53
25	Iris(C)	11	**10**	**10**	**10**	11	95.33	93.33	93.33	93.33	**96.67**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
26	Lymphography(N)	30	31	29	30	**26**	82.14	84.29	83.57	85.00	**86.43**	0.11	0.11	0.11	**0.08**	**0.08**
27	Mushroom(N)L	26	26	**18**	24	24	97.58	99.56	**99.70**	99.56	99.65	10.05	8.91	**5.22**	6.54	7.29
28	Nursery(N)L	358	357	350	**50**	54	67.99	65.38	71.14	**76.85**	74.73	118.36	90.92	82.28	**6.73**	10.83
29	Parkinsons(C)	30	**29**	**29**	**29**	**29**	90.00	**93.33**	92.78	**93.33**	**93.33**	0.13	0.13	**0.12**	0.13	0.13
30	Pendigits(C)L	500	495	460	**456**	487	93.17	94.53	**94.55**	93.85	94.47	248.15	234.61	**170.07**	234.91	171.05
31	Pima-Indians(C)	120	100	100	**68**	75	69.08	70.92	**71.32**	70.13	69.61	0.96	0.67	0.67	**0.36**	0.49
32	P-O-Patient(M)	25	25	23	12	**5**	65.00	68.75	70.00	67.50	**75.00**	0.03	0.03	0.02	0.01	**0.00**
33	Promoters(N)	20	20	18	**14**	16	78.00	84.00	86.00	85.00	**88.00**	0.15	0.15	0.17	**0.05**	0.06
34	Soybean-Large(N)	55	54	**49**	51	52	90.00	90.31	91.41	90.47	**92.03**	1.97	1.77	1.60	**1.42**	1.60
35	Spect(C)	54	53	48	53	**24**	78.46	78.46	78.46	78.46	**82.31**	1.04	1.04	0.93	1.04	**0.29**
36	SPECT-Heart(N)	34	34	32	34	**22**	82.69	82.69	83.08	82.69	**85.00**	0.51	0.46	0.45	0.45	**0.27**
37	Splice(N)L	254	254	237	128	**110**	90.32	90.35	91.22	**93.73**	92.48	173.57	157.26	222.21	**43.45**	60.95
38	Tic-Tac-Toe(N)	28	**27**	**27**	28	**27**	92.74	92.74	**94.42**	92.63	92.74	0.26	0.23	**0.22**	0.24	**0.22**
39	Vehicle(C)	198	197	185	134	**127**	67.20	67.56	67.68	69.15	**70.00**	4.38	4.13	3.88	**1.87**	2.38
40	Wine(C)	29	28	**27**	29	28	87.50	90.63	88.75	**91.88**	90.63	0.05	**0.04**	**0.04**	**0.04**	0.05
Average:		107	105	101	**54**	64	78.93	79.73	80.59	82.11	**82.90**	36.68	32.64	33.21	**12.07**	15.73

Table 3.17 Consolidated Results (continued).

employed in RULES-7 can be made with the help of Table 3.18, which shows the number of datasets for which each of the three new techniques is a winner with respect to the above-mentioned criteria. A graphical representation of this table is also presented in Figures 3.25 – 3.27. It can be seen from the last row in Table 3.17 that the *MS* pruning technique is a winner from the point of view of both *Rules_Reduction* and *Time_Reduction* whereas from *Accuracy_Increase* point of view, the *MPS* pruning technique is clearly the winner, with a cumulative classification accuracy 0.96% higher than *MS*.

3.6 Summary

This chapter has addressed some issues with the RULES-6 algorithm which limited its achievable classification accuracy as well as the reduction in learning time, thereby hampering its ability to handle large noisy datasets. It has also proposed two new pruning techniques derived from association rule learning as well as a novel comparison heuristic in order to boost classification accuracy. Results have proved that the three new techniques introduced into the RULES-7 algorithm make it much more accurate as well as many orders of magnitude faster than its predecessor RULES-6.

No.	New Technique	No. of Datasets for which					
		Min No. of Rules		Max Accuracy		Min Exec. Time	
		Total	%age	Total	%age	Total	%age
1	λ	9	**23%**	10	**25%**	7	**18%**
2	MS	17	**43%**	11	**28%**	23	**58%**
3	MPS	14	**35%**	19	**48%**	10	**25%**

Table 3.18 Summary of All Techniques

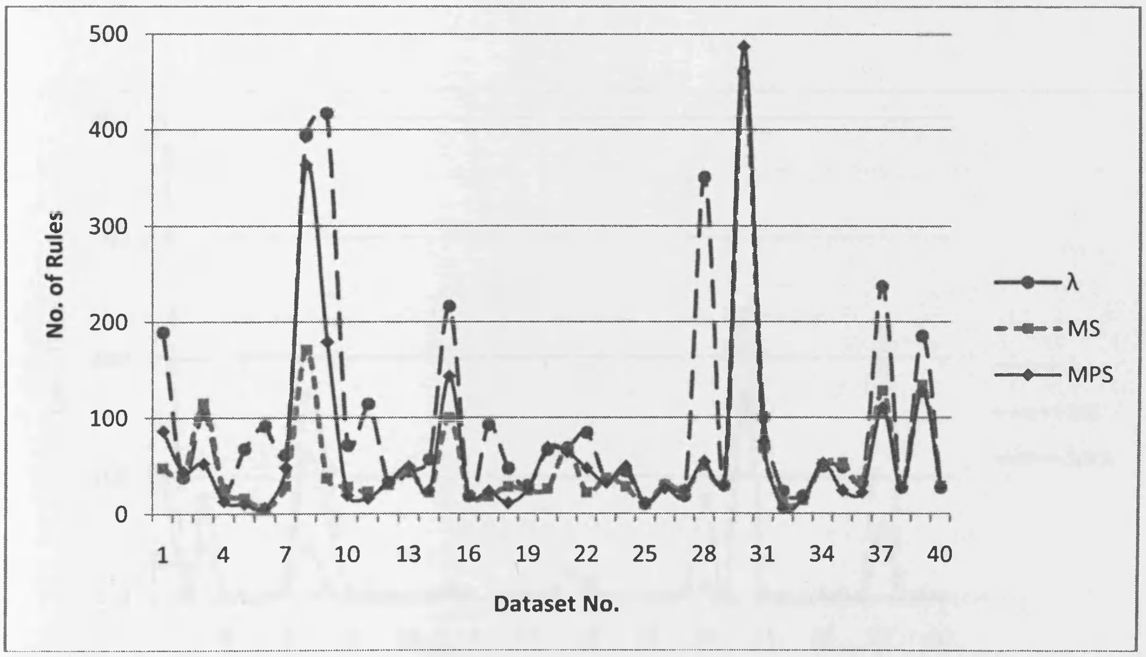


Figure 3.25 RULES-7 New Techniques Comparison in terms of Number of Rules

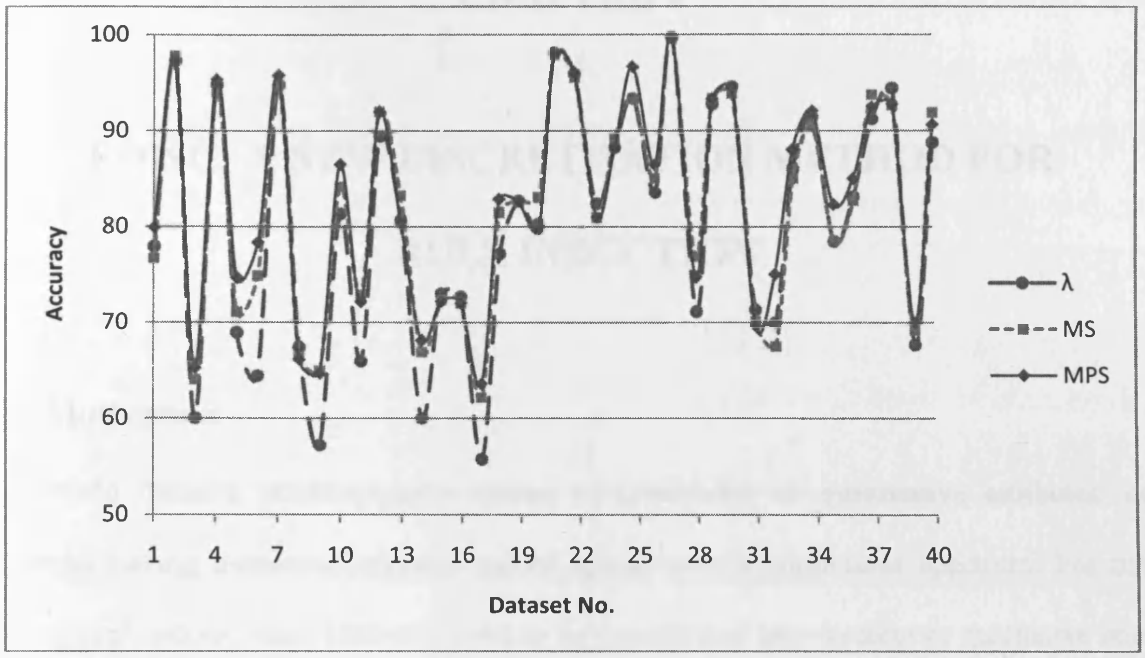


Figure 3.26 RULES-7 New Techniques Comparison in terms of Accuracy

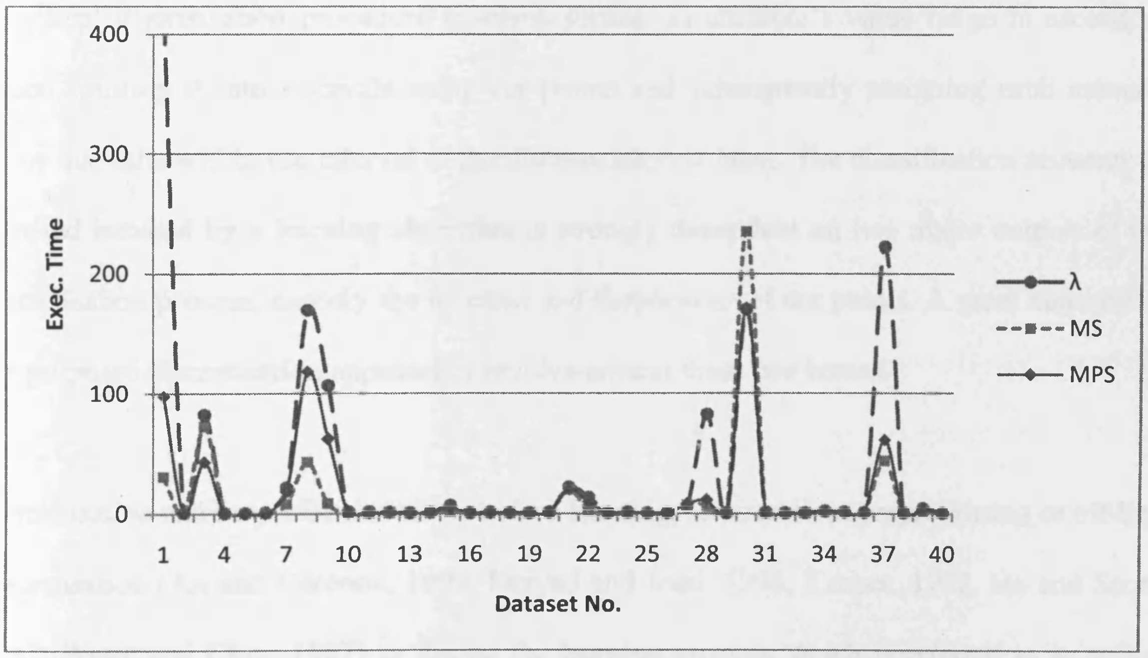


Figure 3.27 RULES-7 New Techniques Comparison in terms of Exec. Time

CHAPTER 4

EDISC: A NEW DISCRETISATION METHOD FOR RULE INDUCTION

4.1 Motivation

Real-world datasets predominantly consist of continuous or quantitative attributes, i.e. attributes having numerous numeric values spread over a continuous spectrum. For data mining applications, such attributes need to be transformed into discrete or qualitative ones. The procedure whereby this transformation is carried out is known as *discretisation*, which has been the focus of active research in the field of data mining for more than a decade.

A typical discretisation procedure involves sorting an attribute's value range in ascending order, splitting it into intervals using cut points and subsequently assigning each numeric value that falls within the interval to the discrete interval label. The classification accuracy of a model induced by a learning algorithm is strongly dependent on two major outputs of the discretisation process, namely the *location* and the *number* of cut points. A great majority of the proposed discretisation approaches revolve around these two issues.

Discretisation can be performed either before learning, referred to as preprocessing or off-line discretisation (An and Cercone, 1999, Fayyad and Irani, 1993, Kerber, 1992, Ho and Scott, 1997, Wong and Chiu, 1987) or during the learning process, which is referred to as online discretisation (Breiman et al., 1984, Clark and Niblett, 1989, Quinlan, 1983, Quinlan, 1986, Quinlan, 1990). The former approach discretises all the continuous attributes before learning, resulting in transformation of the dataset into a discrete one which is then used by the learning algorithm for mining. In the latter approach, the original dataset is passed to the learning algorithm which then discretises its continuous attributes during learning using its built-in discretisation procedure. This involves finding suitable upper and lower bounds (cut points) which mark the boundaries of a discrete interval, so that all continuous attribute values which fall within these two bounds are assigned to that particular discrete interval label.

There are many good reasons in favor of discretisation, particularly preprocessing discretisation, which make it an attractive choice for data mining applications in comparison with online discretisation. Firstly, some induction algorithms are inherently incapable of handling continuous attributes. Others, including most decision tree and rule induction algorithms that can handle numeric attributes, are much slower because of having to sort the attribute values multiple times during the induction process (Witten and Frank, 2005). Secondly, since preprocessing discretisation generalises the raw continuous data, the induction process is faster (Mittal and Cheong, 2002), resulting in a model that is simpler and more accurate (Afify, 2004), as well as more intelligible from the user's point of view (Liu et al., 2002). Finally, automated discretisation in general eliminates the need of a domain knowledge expert to identify the optimal discretisation for a dataset, the extension of which to large datasets is humanly impossible as well as prohibitively expensive for machine learning and data mining applications (Muhlenbach and Rakotomalala, 2005).

The Entropy-MDLP discretisation technique (Fayyad and Irani, 1993) is by and large considered to be the most accurate in the context of both decision tree induction and rule induction algorithms. The basic SRI algorithm (Pham and Afify, 2006a) used this technique as a preprocessing discretisation step. The SRI algorithm works on a class-per-class basis, considers all attribute values to form rule conditions, and removes the covered examples from the dataset each time a new rule is formed. It also employs a stopping criterion in order to decide when to terminate its search for rules. The RULES-6 algorithm is based on SRI with the difference that it uses the RULES family's methodology, which considers the attribute values of the selected seed example only and does not work on a class-per-class basis. Also, RULES-6 does not use any stopping criterion, in contrast to SRI.

The SRI algorithm was later modified to incorporate a new online discretisation technique (Pham and Afify, 2005a) which retained the entropy method but was restricted to a binary splitting approach of decision trees in order to reduce the complexity and runtime. Furthermore, it was found that the full benefits of the search space pruning techniques used in the SRI algorithm could not be realised because the intervals (boundary points) for continuous attributes would change at any location in the rule space. Consequently, instead of using the online discretisation technique of SRI, the RULES-6 algorithm used the standard Entropy-MDLP discretisation as a preprocessing step.

This chapter presents a new discretisation technique EDISC, which uses the Entropy-MDLP method but takes a class-centered approach to discretisation. Since the technique is essentially preprocessing (all the cut points are found prior to learning), it does not have to use the binary splitting approach necessary to reduce complexity during learning. As a result, the discretisation is multi-interval, which is the optimal choice for maximum possible discrimination between the classes. Furthermore, because of finding suitable cut points for each class present in the dataset, the technique is inherently tailored to the seed example approach and consequently results in a significant increase in classification accuracy. Finally, since the boundary points pertain to a specific class, they do not change while processing a particular seed example, thereby resulting in full exploitation of the search space pruning techniques used in the RULES-7 algorithm.

This chapter is organised as follows. Section 4.2 presents a categorisation of the major discretisation approaches found in the literature. Section 4.3 presents a detailed description of the discretisation techniques used for comparison. Section 4.4 outlines the new discretisation technique EDISC. Empirical evaluation of the proposed discretisation technique for the RULES-7 algorithm is presented in section 4.5. Section 4.6 summarises and concludes the chapter.

4.2 Taxonomy of Major Discretisation Approaches

This section is not intended to act as a comprehensive survey of discretisation techniques proposed to date. Because of the existence of diverse discretisation taxonomies in the literature (Dougherty et al., 1995, Kotsiantis and Kanellopoulos, 2006, Yang and Webb, 2005), which in some cases are even conflicting, only the major discretisation categories and representative techniques will be outlined in this section.

4.2.1 Unsupervised vs. Supervised

Unsupervised discretisation techniques do not take into account the class labels of the values of the continuous attribute being discretised (Dougherty et al., 1995, Pham and Afify, 2005a). Typical examples of this approach are Equal-width and Equal-frequency discretisation (Wong and Chiu, 1987). Both the techniques require a user-specified input parameter, which is the number of intervals in the former one and the number of values within each interval in the latter. Since the input parameter is quite arbitrary, the efficacy of both the techniques varies. Supervised discretisation techniques on the other hand take the class labels of the continuous attribute values into account when forming intervals. Techniques such as 1R (Holte, 1993), Entropy-MDLP (Fayyad and Irani, 1993) etc. are examples of this approach.

4.2.2 Static vs. Dynamic

This is also referred to in the literature as univariate vs. multivariate (Bay, 2000). In case of static discretisation, each continuous attribute in the dataset is considered independently of the other continuous attributes for the purpose of forming intervals. Equal-width (Wong and Chiu, 1987), 1R (Holte, 1993), as well as Entropy-MDLP (Fayyad and Irani, 1993) discretisation techniques fall into this category. The dynamic discretisation approach takes all the continuous attributes into account so that the intervals are formed keeping in view the interdependencies of attributes (Bay, 2000, Gama et al., 1998). Little work has been done in the area of dynamic discretisation and a great majority of the existing discretisation techniques are static.

4.2.3 Global vs. Local

This is also referred to in the literature as *Offline or Preprocessing vs. Online Discretisation* (Kotsiantis and Kanellopoulos, 2006, Pham and Afify, 2005a). The global discretisation approach transforms the original dataset into a discrete one by discretising all the continuous attributes in it prior to the start of the learning phase. For example, the Equal-width (Wong and Chiu, 1987) algorithm is solely a preprocessing discretisation technique. The local discretisation approach instead operates on a subset of the instance space during learning. The Entropy-MDLP (Fayyad and Irani, 1993) discretisation method is used as a global discretisation technique, whereas the minimal entropy binary splitting method is used as a local discretisation technique in C4.5 (Quinlan, 1993). Global discretisation techniques have empirically proven to be more effective than local ones because they use the attribute's entire domain range instead of operating on a subpartition of it (Kotsiantis and Kanellopoulos, 2006).

4.2.4 Parametric vs. Non-parametric

Parametric discretisation techniques require one or more input parameters from the user whereas non-parametric techniques do not (Pham and Afify, 2005a). An example of unsupervised parametric discretisation is the Equal-width (Wong and Chiu, 1987) technique, which requires the user to input the number of intervals. Entropy-MDLP (Fayyad and Irani, 1993) discretisation, on the other hand, can be classified as a supervised non-parametric method.

4.2.5 Hierarchical vs. Non-hierarchical

Hierarchical discretisation techniques follow the general-to-specific approach (Yang and Webb, 2005). This means that they initially consider the whole value range of the attribute as one interval and then successively split it into subintervals until the termination condition is satisfied. By contrast, non-hierarchical techniques adopt the specific-to-general approach by initially considering all the values within the range as intervals and then repeatedly merging them until the termination condition is satisfied. StatDisc (Richeldi and Rossotto, 1995) and ChiMerge (Kerber, 1992) represent the former and the latter approach respectively. The limitation of both these methods is the requirement of a user-specified input parameter called the significance level.

4.2.6 Disjoint vs. Non-disjoint

Disjoint discretisation techniques are characterised by non-overlapping intervals whereas non-disjoint techniques allow overlapping between the intervals (Yang and Webb, 2005). The discretisation technique proposed in this work falls in the latter category.

4.2.7 Fuzzy vs. Non-fuzzy

In the case of fuzzy discretisation, the intervals are not strict so that a value may belong to multiple intervals each with a certain degree of membership. This is not the case with non-fuzzy discretisation, where a value may belong to strictly one interval (Yang and Webb, 2005).

4.3 Description of Techniques Used for Comparison

Three techniques have been used as preprocessing discretisation procedures with RULES-7 for the purpose of comparison with the new technique. All the techniques are univariate, i.e. they operate on each attribute independently after sorting its values in ascending order. A detailed description of the techniques is as follows.

4.3.1 Equal-width Discretisation

The Equal-width (Wong and Chiu, 1987) discretisation technique determines the interval width according to the user-specified number of intervals using the relation:

$$i_w = (max\_value - min\_value) / n \qquad (4.1)$$

where

i_w = interval width

max_value = maximum value of the attritbute

min_value = minimum value of the attritbute

n = user-specified number of intervals

It then creates the cut points using the relation:

$$cut\_point_i = min\_value + j \times i_w \tag{4.2}$$

where

$j = 1, 2, \ldots.. n - 1$

4.3.2 1R Discretisation

The 1R discretisation technique (Holte, 1993) works on the principle of majority class so that a certain discretisation interval is dominated by instances of a particular class. In order to ensure that the intervals do not become overly specific, an interval must contain at least a certain minimum number of instances (attribute values) as per the description of 1R discretiser in the literature. However, since almost all the datasets are characterised by multiple duplicate values for an attribute, this translates to a certain minimum number of unique attribute values per interval. The original 1R discretiser set this parameter at 6.

While scanning the attribute values, the counts of each class are updated as the scan progresses. As soon as the count of the unique attribute values reaches 6, the majority class is determined and the scan continues as long as the next attribute values have the same class. A cut point is placed at the attribute value whose class is different from the majority class. Finally, adjacent intervals with the same class labels are merged.

4.3.3 Entropy-MDLP Discretisation

The discretisation technique in the decision tree algorithm ID3 (Quinlan, 1986) uses the entropy measure from information theory (Shannon and Weaver, 1963, Thornton, 1992). The measure was also later employed to handle continuous attributes in the decision tree algorithm C4.5 (Quinlan, 1993). The entropy measure in the context of classification can be defined as (Liu et al., 2002):

$$E_c = E_1 + E_2 \tag{4.3}$$

$$E_c = -\,p_{left}\sum_{i=1}^{k} p_{i,left}\log p_{i,left} - p_{right}\sum_{i=1}^{k} p_{i,right}\log p_{i,right} \tag{4.4}$$

where

E_c = entropy of the cut-point

E_1 = entropy to the left of the cut-point

E_2 = entropy to the right of the cut-point

k = total number of classes

i = a particular class

$$p_{left} = \frac{\text{number of instances to the } left \text{ of cut-point}}{\text{total number of instances, } N}$$

$$p_{right} = \frac{\text{number of instances to the } right \text{ of cut-point}}{\text{total number of instances, } N}$$

$$p_{i,left} = \frac{\text{number of instances of } Class\ i \text{ to the } left \text{ of cut-point}}{\text{number of instances to the } left \text{ of cut-point}}$$

$$p_{i,right} = \frac{\text{number of instances of } Class\ i \text{ to the } right \text{ of cut-point}}{\text{number of instances to the } right \text{ of cut-point}}$$

The decision tree induction algorithms mentioned above use the entropy measure at each node of the tree and split at the point with the minimum entropy. All unique attribute values are considered as candidate split points and splitting continues until the termination condition is satisfied, i.e. all the instances in a leaf node belong to a single class.

Later, a new preprocessing discretisation technique (Fayyad and Irani, 1993) was proposed which showed that it was necessary to examine only attribute values where the class changes, referred to as the *boundary cut points*. This significantly improved efficiency because of restricting the search space. It was also suggested that binary splitting was suboptimal, so the splitting process should continue recursively to get multiple discretisation intervals. The stopping criterion was based on the MDLP principle which is defined as (Witten and Frank, 2005):

$$gain > \frac{log_2(N-1)}{N} + \frac{log_2(3^k-2) - kE + k_1E_1 + k_2E_2}{N} \tag{4.5}$$

where

$$E = -\sum_{i=1}^{k} p_i \, log \; p_i \,, \qquad p_i = \frac{\text{number of instances of } Class\ i}{\text{total number of instances, } N}$$

$gain = E - E_c =$ information gained by splitting at the cut-point

$N =$ total number of instances in the attribute value list at each recursion

$k_1 =$ number of classes to the left of the cut-point

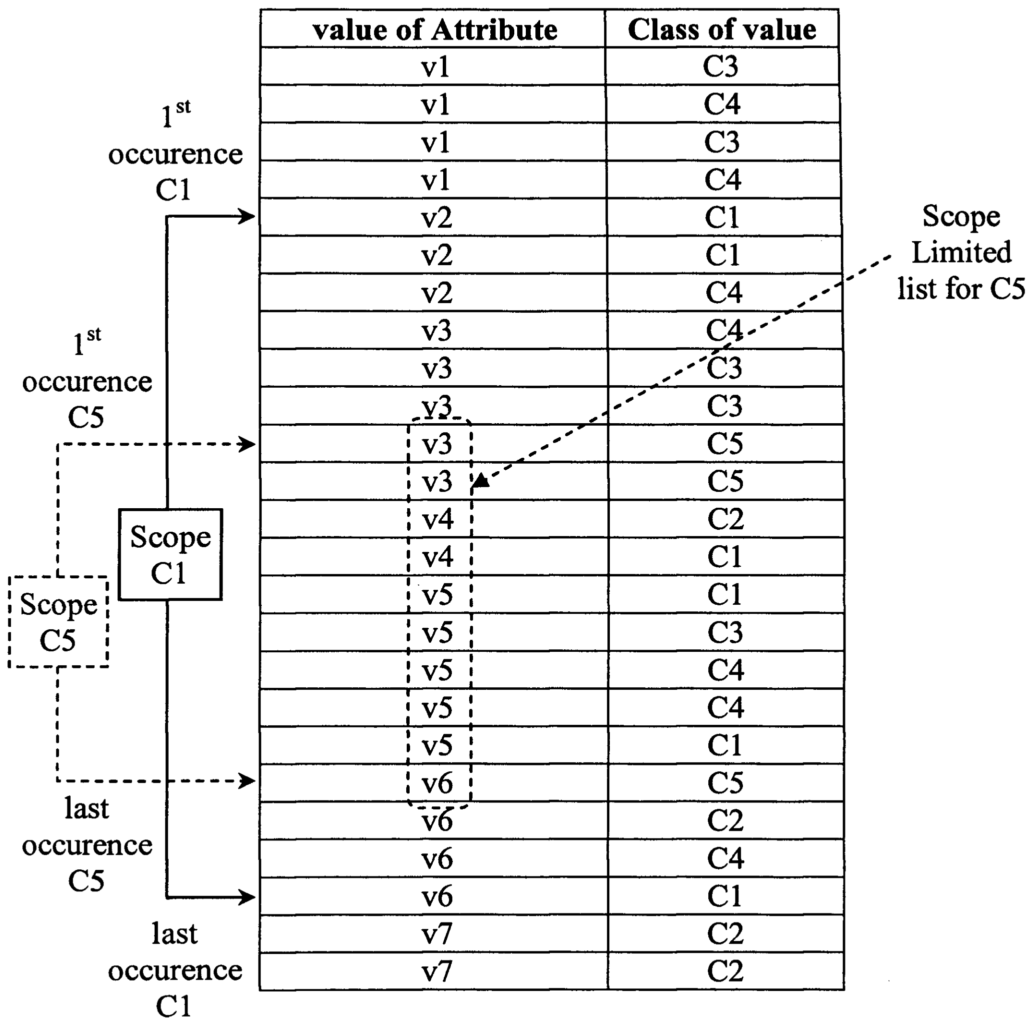
$k_2 =$ number of classes to the right of the cut-point

4.4 Proposed Discretisation Technique

The proposed discretisation technique has been named EDISC, which stands for "Entropy-based Discretisation Intervals using Scope of Classes". It makes use of a critical observation regarding the occurrence of classes in the datasets employed in this work. It has been found that among the sorted continuous values of any attribute within a dataset containing *k* classes, the starting and ending points for the *i*th class ($i \in \{1, 2, 3 \ldots k\}$) seldom occur in the minimum and maximum values of the continuous attribute respectively. In fact, different classes start and end in different continuous value blocks within the range of sorted values. This can be clarified with the help of Figure 4.1 which shows a typical dataset containing 5 classes and having 7 values for a particular attribute.

It is clear from Figure 4.1 that the first occurrence of class C1 is in the value v2 while the last occurrence is in the value v6, so the *scope* of C1 is v2–v6. Similarly the scope of C2, C3, C4 and C5 is v4–v7, v1–v5, v1–v6 and v3–v6 respectively. The traditional Entropy-MDLP discretisation technique identifies the cut points for each continuous attribute within the dataset without taking into consideration whether the cut points are optimal for each class present. Since the goal in classification rule induction tasks is to maximise the coverage of the positive class along with minimising the coverage of the negative classes, such cut points are not optimal for a particular class. The new discretisation technique EDISC addresses this issue by using the concept of the scope of classes as defined above. The EDISC algorithm is outlined in Figure 4.2 and its operation is as follows.

After selecting a continuous attribute to be discretised, the values of the attribute are copied from the instance list onto a user-defined data type *value_class_list,* which also includes a *class_label* data member as well as an *on_boundary* indicator. At steps 4-5, the *value_class_list* is sorted in ascending order and boundary values are identified. At steps 6-7, the scope of a particular class present in the dataset is determined in the *value_class_list* and subsequently cut points are found in this *scope* limited *value_class_list* using the usual Entropy-MDLP method at step 8. A cut point is taken as the middle value between an *on_boundary* value and the next value. The minimum and maximum values of the attribute



value of Attribute	Class of value
v1	C3
v1	C4
v1	C3
v1	C4
v2	C1
v2	C1
v2	C4
v3	C4
v3	C3
v3	C3
v3	C5
v3	C5
v4	C2
v4	C1
v5	C1
v5	C3
v5	C4
v5	C4
v5	C1
v6	C5
v6	C2
v6	C4
v6	C1
v7	C2
v7	C2

Figure 4.1 Illustration of the concept of *Scope* of Classes

```
Procedure EDISC (Dataset)
For each Attribute present in the DataSet Do                                  (step 1)
  If the Attribute is Continuous Then                                         (step 2)
    Copy the Attribute values along with Class labels to value_class_list     (step 3)
    Sort the value_class_list in value ascending order                        (step 4)
    Find the boundary values in the value_class_list                          (step 5)
    For each Class present in the DataSet Do                                  (step 6)
      Find the Scope of the Class in the value_class_list                     (step 7)
      discrete_table.Attribute.Class.cut_points = Entropy_MDLP (Scope, value_class_list)
                                                                              (step 8)
Return discrete_table                                                         (step 9)
```

Figure 4.2 The EDISC Algorithm

for the scope limited list are also added to the final cut point list for each class but are obviously not to be counted as cut points. After cut points for all the classes have been found for each attribute, the discrete_table which stores all the discretisation information is returned at step 9. A comparison of the cut points discovered by the new discretisation technique EDISC against those found by the standard Entropy-MDLP discretisation technique for the datasets *Iris(C)*, *Cover-Type(M)L*, *Landsat(C)L* and *Letter(C)L* is shown in Tables 4.1 – 4.4 respectively.

To handle continuous attributes using the new technique, both the RULES-7 algorithm as well as the *Induce_One_Rule* procedure need to be modified slightly, as shown in Figure 4.3 and Figure 4.4 respectively. Additionally, a *Find_Interval* procedure is required at step 1 in the *Induce_One_Rule* procedure to determine the upper and lower bounds for the conditions in the antecedent of the *BestRule* which correspond to the Attribute-values in the seed example. The *Find_Interval* procedure is outlined in Figure 4.5. Upon entering the RULES-7 algorithm, the EDISC procedure is called at step 2 which returns the discrete_table on completion. This is assigned to the *Disc_Table* which is then passed as an argument to the *Induce_One_Rule* procedure at step 6. Here, the *Find_Interval* procedure is called at step 1 which returns a rule with conditions corresponding to the attribute values of the seed example but having the upper and lower bounds set, so that the value becomes a discrete one by being replaced with the interval so defined. The rest of the *Induce_One_Rule* procedure remains unmodified.

4.5 Empirical Evaluation of EDISC

This section presents the results obtained from the experimental evaluation of EDISC against the three discretisation techniques outlined in section 4.3. In order to ensure that the

Attribute #	Technique	Class #	Cut-Points	# of Cuts
1.	*Entropy_MDLP*	-	4.3 5.55 6.15 7.9	2
	EDISC	1	4.3 5.45 5.8	1
		2	4.9 5.55 7	1
		3	4.9 5.55 6.15 7.9	2
2.	*Entropy_MDLP*	-	2 3.35 4.4	1
	EDISC	1	2.3 3.35 4.4	1
		2	2 2.95 3.4	1
		3	2.2 3.05 3.8	1
3.	*Entropy_MDLP*	-	1 2.45 4.75 6.9	2
	EDISC	1	1 1.9	0
		2	3 4.75 5.1	1
		3	4.5 4.95 6.9	1
4.	*Entropy_MDLP*	-	0.1 0.8 1.75 2.5	2
	EDISC	1	0.1 0.6	0
		2	1 1.65 1.8	1
		3	1.4 1.75 2.5	1

Table 4.1 EDISC Comparison with Entropy_MDLP for *Iris* Data

Technique	Class #	Cut-Points for Attribute 3	# of Cuts
Entropy_MDLP	-	0 13.5 19.5 26.5 66	3
EDISC	1	1 13.5 19.5 26.5 38	3
	2	0 13.5 19.5 26.5 45	3
	3	1 13.5 19.5 26.5 44	3
	4	4 18.5 29	1
	5	4 18.5 26	1
	6	9 13.5 22.5 32.5 52	3
	7	2 13.5 19.5 27.5 39	3

Table 4.2 EDISC Comparison with Entropy_MDLP for *Cover-Type* Data

Technique	Class #	Cut-Points for Attribute 1	# of Cuts
Entropy_MDLP	-	39 48.5 51.5 58 61.5 66.5 71.5 77 82.5 86.5 104	9
EDISC	1	46 48.5 49.5 52.5 58 61.5 66.5 71.5 77 82.5 84.5 97	10
	2	39 48.5 51.5 58 61.5 69 71.5 75.5 81 84.5 89	9
	3	56 61.5 66.5 71.5 77 82.5 86.5 104	6
	4	46 48.5 49.5 52.5 58 61.5 66.5 71.5 77 82.5 84.5 97	10
	5	43 48.5 51.5 58 61.5 69 71.5 75.5 81 84.5 92	9
	6	47 48.5 51.5 58 61.5 66.5 71.5 77 82.5 84.5 95	9

Table 4.3 EDISC Comparison with Entropy_MDLP for *Landsat* Data

Technique	Class #	Cut-Points for Attribute 3	# of Cut-Points
Entropy_MDLP	-	0 0.5 3.5 7.5 9.5 15	4
EDISC	1	2 3.5 6.5 8.5 11	3
	2	1 3.5 6.5 8.5 10	3
	3	1 2.5 4.5 7.5 9	3
	4	1 3.5 6.5 8.5 10	3
	5	1 3.5 6.5 8.5 10	3
	6	1 3.5 6.5 8.5 10	3
	7	1 3.5 6.5 8.5 10	3
	8	1 3.5 6.5 8.5 12	3
	9	0 0.5 1.5 3.5 6.5 9	4
	10	1 2.5 4.5 7.5 9	3
	11	1 3.5 6.5 8.5 11	3
	12	1 2.5 4.5 7.5 9	3
	13	1 3.5 6.5 8.5 15	3
	14	1 3.5 6.5 8.5 14	3
	15	1 3.5 6.5 8.5 10	3
	16	1 3.5 6.5 8.5 10	3
	17	1 3.5 6.5 8.5 11	3
	18	1 3.5 6.5 8.5 10	3
	19	1 3.5 6.5 8.5 10	3
	20	1 2.5 4.5 7.5 9	3
	21	1 3.5 6.5 8.5 10	3
	22	1 3.5 6.5 8.5 10	3
	23	1 3.5 6.5 8.5 13	3
	24	1 3.5 6.5 8.5 11	3
	25	1 3.5 6.5 8.5 10	3
	26	1 3.5 6.5 8.5 10	3

Table 4.4 EDISC Comparison with Entropy_MDLP for *Letter* Data

```
Procedure RULES-7 (Dataset, w, MinSup, MinPosSup, λ)
RuleSet = ∅                                                                  (step 1)
Disc_Table = EDISC (DataSet)                                                 (step 2)
Compute the no. of instances a ParentRule should cover as per MinSup         (step 3)
While there are instances in the Dataset that have not been covered Do       (step 4)
  Take a seed example s that has not yet been covered.                       (step 5)
  Rule = Induce_One_Rule (s, DataSet, Disc_Table, w, MinSup, MinPosSup, λ)  (step 6)
  Mark the instances covered by the Rule as covered.                         (step 7)
  Add Rule to the RuleSet                                                    (step 8)
End While
Return RuleSet                                                               (step 9)
End                                                                          (step 10)
```

Figure 4.3 Modified pseudo-code description of RULES-7

```
Procedure Induce_One_Rule (s, DataSet, Disc_Table, w, MinSup, MinPosSup, λ)
ParentRuleSet = ChildRuleSet = ∅
BestRule = Find_Interval (s, Disc_Table)                                    (step 1)
BestRule = rule with no condition (empty antecedent)
Compute the no. of instances a ChildRule should cover within the target class as per
MinPosSup                                                                   (step 2)
Add BestRule to the ParentRuleSet                                           (step 3)
While ParentRuleSet is NOT Empty Do                                         (step 4)
 For each ParentRule ∈ ParentRuleSet Do
  If ParentRule.Covered ≥ MinSupport Then                                   (step 5)
   For each nominal Attribute $A_i$ Do
    If $Condition_i$ ∉ ParentRule AND $Condition_i$ is not marked as "Invalid" for ParentRule
    Then
      ChildRule = ParentRule ∧ $Condition_i$                                 (step 6)
     If ChildRule ∉ ChildRuleSet Then                                       (step 7)
     {Check the ChildRuleSet for duplicate rule of the candidate ChildRule}
      Compute ChildRule.Instances
      If ChildRule.Classified ≥ MinPosSupport AND                           (step 8)
        ParentRule.Misclassified − ChildRule.Misclassified ≥ MinExcludedNeg Then
                                                                            (step 9)
       Compute ChildRule.Quality
       If (ChildRule.Score > BestRule.Score) OR
         (ChildRule.Score > λ × BestRule.Score AND
          ChildRule.NewClassified > BestRule.NewClassified) Then            (step 10)
         BestRule = ChildRule
       If ChildRule.Consistency ≠ 100% Then                                 (step 11)
         Add ChildRule to ChildRuleSet                                      (step 12)
       Else
        Mark $Condition_i$ as "Invalid" for ParentRule                       (step 13)
Else
```

Figure 4.4 Modified pseudo-code description of the *Induce_One_Rule* procedure of RULES-7

```
            Mark Condition$_i$ as "Invalid" for ParentRule                      (step 14)
        End For
    End For
    Empty ParentRuleSet
    For each ChildRule $\in$ ChildRuleSet Do
      Compute ChildRule.OptimisticScore
      If ChildRule.OptimisticScore $\leq$ BestRule.Score Then                    (step 15)
        Delete the ChildRule from the ChildRuleSet                             (step 16)
        For each nominal Attribute $A_i$ Do
          If Condition$_i$ $\in$ ChildRule AND Condition$_i$ $\notin$ ParentRule Then    (step 17)
            Mark Condition$_i$ as "Invalid" for ParentRule                     (step 18)
            Exit For Loop
        End For
    End For

    For each ChildRule $\in$ ChildRuleSet Do
    {Mark any Conditions of remaining ChildRules of each ParentRule as Invalid which are
      $\notin$ ParentRule and "marked as Invalid" for the ParentRule}
      For each nominal Attribute $A_i$ Do
        If Condition$_i$ $\in$ ChildRule AND Condition$_i$ $\notin$ ParentRule Then      (step 19)
          If Condition$_i$ is marked as "Invalid" for ParentRule Then          (step 20)
            Mark Condition$_i$ as "Invalid" for ChildRule                      (step 21)
      End For
    End For
    If $w > 1$ Then
      Add $w$ highest Score ChildRules from ChildRuleSet into ParentRuleSet     (step 22)
      Empty ChildRuleSet
  End While
  Return BestRule
  End
```

Figure 4.4 Modified pseudo-code description of the *Induce_One_Rule* procedure of RULES-7 (continued).

```
Procedure Find_Interval (seed, Disc_Table)
For each Attribute present in the Seed Do                                   (step 1)
  If the Attribute is Continuous Then                                       (step 2)
    Class = seed.class
    Scan Disc_Table.Attribute.Class.cut_points list to find the upper_bound  (step 3)
    and lower_bound within which the Attribute.value falls
    Rule.Condition.upper_bound = upper_bound                                (step 4)
    Rule.Condition.lower_bound = lower_bound                                (step 5)
Return Rule
```

Figure 4.5 The *Find_Interval* procedure

evaluation is as comprehensive as possible, experiments were conducted on 35 benchmark datasets shown in Table 4.5. All these datasets were downloaded from the University of California at Irvine (UCI) repository of machine learning databases (Blake and Merz, 1998), except the *Depression* dataset which was obtained from Williams College College (Veaux, 2007). Of these 35 datasets, 22 are continuous whereas the rest are mixed type. Each dataset name is followed by either (C) or (M), indicating that the dataset is continuous or mixed respectively, and occasionally by the suffix L, indicating that the dataset is large. All the datasets contain the full sample size except the *Cover-Type(M)L* dataset which was sampled to 1/187 of its original size in order to finish processing within a reasonable amount of time.

All tests were conducted on an Intel Pentium 2.0 GHz Dual-Core computer with 2 GB of RAM and Windows XP operating system. In order to make sure that the prediction estimate is accurate, the evaluation approach used in this study is the standard *stratified 10-fold cross-validation* (Kohavi, 1995a, Witten and Frank, 2005). To evaluate the performance of the discretisation techniques, four criteria were used namely *number of rules*, *accuracy*, *discretisation time* and *execution time*, of which the second and third criteria are of prime importance for the purpose of evaluating a discretisation technique. All execution times are reported in seconds. An in-depth comparison of the new discretisation technique EDISC against the three other techniques is shown in Table 4.6 with the best values formatted bold, whereas a summary in terms of the number of datasets for which a technique has been effective is presented in Table 4.7.

It is clear from the last row in Table 4.6 that EDISC is a winner from the point of view of not only the most important criteria *accuracy* and *discretisation time* but also *execution time*. Although the Equal-width discretisation technique emerges as the winner in terms of the total

No.	Dataset	No. of Instances	No. of Attributes			No. of Classes (k)
			Total	Categorical	Continuous	
1	Adult(M)L	48,842	14	8	6	2
2	Anneal(M)	798	38	32	6	6
3	Arrhythmia(M)L	452	279	73	206	13
4	Blood-Transfusion(C)	748	4	0	4	2
5	Breast-Cancer(C)	699	10	0	10	2
6	Connectionist-Bench(C)	208	60	0	60	2
7	*Cover-Type(M)L	3,100	54	47	7	7
8	Credit-Approval(M)	690	15	9	6	2
9	Cylinder-Bands(M)	541	39	19	20	2
10	Depression(M)	428	17	11	6	2
11	Dermatology(M)	366	34	33	1	6
12	Echocardiogram(M)	132	12	4	8	2
13	Ecoli(C)	336	8	0	8	8
14	Glass(C)	214	10	0	10	6
15	Heart-Cleveland(M)	303	13	8	5	5
16	Heart-Hungarian(M)	294	13	8	5	2
17	Hepatitis(M)	155	19	13	6	2
18	Hypothyroid(M)L	3,772	29	22	7	2
19	Image(C)	210	19	0	19	7
20	Ionosphere(C)	351	34	0	34	2
21	Iris(C)	150	4	0	4	3
22	Landsat(C)L	6,435	36	0	36	6
23	Letter(C)L	20,000	16	0	16	26
24	Magic(C)L	19,020	10	0	10	2
25	Optical(C)L	5,620	64	0	64	10
26	Ozone(C)L	2,534	72	0	72	2
27	Page-Blocks(C)L	5,473	10	0	10	5
28	Pendigits(C)L	10,992	16	0	16	10
29	Shuttle(C)L	58,000	5	0	5	7
30	Spambase(C)L	4,601	57	0	57	2
31	Spect(C)	267	44	0	44	2
32	Vehicle(C)	846	18	0	18	4
33	Waveform-v2(C)L	5,000	40	0	40	3
34	Wine(C)	178	13	0	13	3
35	Yeast(C)	1,484	8	0	8	10

Table 4.5 Summary of Datasets used in Experiments

No.	Dataset	No. of Rules				Accuracy (%)				Disc. Time (sec)				Exec. Time (sec)			
		EDISC	EW	1R	ENT	EDISC	EW	1R	ENT	EDISC	EW	1R	ENT	EDISC	EW	1R	ENT
1	Adult(M)L	196	**173**	1076	681	83.17	78.18	83.57	**85.05**	**0.19**	1.19	2.05	1.36	323.15	**243.40**	1403.41	1278.09
2	Anneal(M)	33	44	**31**	33	**98.18**	97.79	91.43	97.66	**0.00**	0.06	0.09	0.11	**0.35**	0.51	0.36	**0.35**
3	Arrythmia(M)L	**71**	111	113	104	**74.00**	60.00	66.00	67.25	0.45	0.44	0.52	**0.41**	101.50	163.94	87.34	**49.56**
4	Blood-Transfusion(C)	**1**	2	3	3	**77.03**	**77.03**	**77.03**	**77.03**	**0.00**	0.03	0.05	0.03	**0.00**	0.01	0.01	0.01
5	Breast-Cancer(C)	25	27	17	**16**	93.48	**94.06**	93.19	93.04	**0.02**	0.06	0.06	0.08	0.11	0.09	0.11	**0.07**
6	Connectionist (C)	**28**	47	53	30	**83.00**	66.00	49.50	81.00	**0.05**	0.08	0.16	0.16	0.69	0.89	**0.25**	0.45
7	Cover-Type(M)L	446	**445**	548	456	**58.34**	56.16	56.09	55.90	**0.13**	0.28	0.22	0.22	**107.84**	114.62	130.03	110.08
8	Credit-Approval(M)	77	82	84	**59**	81.91	81.03	**82.50**	80.88	**0.00**	0.05	0.06	0.08	1.40	1.30	**0.99**	1.37
9	Cylinder-Bands(M)	**88**	109	108	114	**61.89**	61.51	59.62	61.13	**0.05**	0.13	0.14	0.11	4.86	3.88	4.20	**3.44**
10	Depression(M)	**58**	117	111	123	67.86	67.38	**69.52**	65.95	**0.05**	**0.05**	0.06	0.09	**0.75**	1.03	0.99	0.97
11	Dermatology(M)	**35**	36	**35**	**35**	**90.00**	89.43	89.14	**90.00**	**0.00**	0.02	0.05	0.02	**0.71**	0.73	0.76	0.76
12	Echocardiogram(M)	**11**	21	22	17	**57.50**	52.50	55.00	54.17	**0.00**	0.05	0.02	0.02	**0.03**	**0.03**	**0.03**	**0.03**
13	Ecoli(C)	23	52	71	**16**	**85.81**	78.39	69.03	81.94	**0.00**	0.03	0.03	0.03	0.05	0.08	0.10	**0.04**
14	Glass(C)	**18**	38	72	33	**72.78**	55.56	65.00	69.44	**0.02**	0.03	0.03	0.05	**0.05**	0.06	0.08	0.07
15	Heart-Cleveland(M)	**69**	97	97	100	56.07	55.36	**57.86**	53.93	**0.02**	**0.02**	**0.02**	**0.02**	0.46	0.51	**0.41**	0.45
16	Heart-Hungarian(M)	**25**	47	48	36	**80.36**	77.86	76.07	76.79	**0.00**	0.02	0.09	0.02	**0.12**	0.18	0.18	**0.12**
17	Hepatitis(M)	**24**	30	30	26	**84.00**	81.33	82.67	**84.00**	**0.00**	**0.00**	0.03	0.02	0.10	0.11	0.09	**0.08**
18	Hypothyroid(M)L	**41**	119	74	47	**99.02**	85.11	98.06	98.78	**0.03**	0.20	0.28	0.20	9.79	46.39	14.75	**8.92**
19	Image(C)	24	38	44	**21**	**91.43**	79.05	77.62	90.00	**0.03**	0.05	0.06	0.06	0.10	0.16	**0.07**	0.08
20	Ionosphere(C)	**33**	47	48	36	**92.06**	86.18	86.76	91.47	**0.05**	0.11	0.14	0.14	0.41	0.48	**0.20**	0.39

Table 4.6 Comparison of EDISC Against Three Other Discretisation Techniques

No.	Dataset	No. of Rules				Accuracy (%)				Disc. Time (sec)				Exec. Time (sec)			
		EDISC	EW	1R	ENT	EDISC	EW	1R	ENT	EDISC	EW	1R	ENT	EDISC	EW	1R	ENT
21	Iris(C)	6	11	**5**	**5**	93.33	95.33	**96.00**	91.33	0.03	**0.00**	0.03	0.02	**0.00**	**0.00**	**0.00**	**0.00**
22	Landsat(C)L	647	**458**	661	635	**85.36**	84.50	82.51	83.97	**0.34**	1.25	1.38	1.41	204.38	295.28	226.15	**192.76**
23	Letter(C)L	2319	2003	**567**	2561	**82.10**	80.21	42.20	79.14	**0.73**	1.56	1.63	1.78	1832.67	2353.99	**819.26**	1914.69
24	Magic(C)L	599	**204**	3397	601	**79.64**	76.76	74.13	79.63	**0.73**	0.99	4.53	1.53	236.76	**87.28**	371.43	266.98
25	Optical(C)L	468	447	**370**	477	89.69	88.89	**90.74**	88.28	**0.47**	1.52	1.56	1.56	472.45	416.33	452.43	**405.59**
26	Ozone(C)L	93	**91**	96	94	**93.72**	**93.72**	**93.72**	**93.72**	**0.14**	0.91	1.03	0.94	22.51	19.33	**11.99**	25.28
27	Page-Blocks(C)L	88	**34**	323	92	92.50	91.51	92.44	**92.83**	**0.16**	0.42	0.38	0.41	6.95	**2.22**	12.90	5.95
28	Pendigits(C)L	725	**497**	799	724	92.08	**93.35**	88.01	90.14	**0.39**	0.91	1.02	1.03	**189.09**	217.07	225.37	204.94
29	Shuttle(C)L	99	**32**	87	102	**99.83**	90.35	99.76	99.81	**0.58**	2.95	3.05	3.06	44.90	**23.23**	43.19	45.51
30	Spambase(C)L	157	**71**	231	158	**90.54**	76.04	88.82	89.93	**0.22**	0.89	1.09	1.13	132.76	406.57	289.66	**131.77**
31	Spect(C)	**15**	53	46	47	**82.31**	78.46	81.92	80.00	**0.03**	0.14	0.14	0.14	0.75	1.09	0.80	**0.55**
32	Vehicle(C)	130	200	214	**126**	**68.78**	67.20	60.98	68.54	**0.05**	0.14	0.16	0.17	3.76	4.08	**3.00**	3.35
33	Waveform-v2(C)L	718	**709**	1117	724	**80.89**	78.53	57.30	79.89	**0.36**	1.03	1.20	1.03	168.63	296.41	**142.35**	153.17
34	Wine(C)	**7**	28	20	8	97.50	87.50	90.63	**98.13**	**0.02**	0.03	0.03	0.03	**0.01**	0.04	0.02	0.02
35	Yeast(C)	**49**	124	410	121	**55.17**	46.62	45.66	53.66	**0.05**	0.11	0.13	0.11	**0.65**	1.40	3.72	1.50
	Average:	213	**190**	315	242	**82.04**	77.40	76.30	80.70	**0.15**	0.45	0.61	0.50	**110.54**	134.36	121.33	137.35

Table 4.6 Comparison of EDISC Against Three Other Discretisation Techniques (continued).

No.	Discretisation Technique	No. of Datasets for which							
		Min No. of Rules		Max Accuracy		Min Disc. Time		Min Exec. Time	
		Total	%age	Total	%age	Total	%age	Total	%age
1	EDISC	16	46%	25	71%	33	94%	12	34%
2	EW	10	29%	2	6%	1	3%	4	11%
3	1R	4	11%	5	14%	0	0%	9	26%
4	ENT	5	14%	3	9%	1	3%	10	29%

Table 4.7 Summary of All Discretisation Techniques

number of rules for all the datasets, it is clear from Table 4.7 that this is not the case. The EDISC discretisation technique is also a winner from the point of view of the *number of rules* criterion. This is because EDISC gave minimum rules for 16 out of 35 datasets, which equates to 46% of the total number of datasets employed. The significant difference in the total number of rules for the two techniques is only because of 5 datasets listed as numbers 22-24, 28 and 30. Furthermore, the cumulative classification accuracy of the Equal-width technique is much less than that of EDISC. This is also evident from Table 4.7, which shows that the Equal-width discretisation technique gave the maximum accuracy for only 2 datasets out of 35, whereas EDISC resulted in maximum accuracy for 25 datasets. The most visible increase in accuracy with EDISC (greater than 1.5%) compared to Entropy-MDLP can be seen for 12 datasets listed as no. 3, 6, 7, 12-14, 16, 19, 22, 23, 31 and 35. The top 5 datasets in terms of an accuracy boost are *Arrythmia(M)L, Echocardiogram(M), Glass(C), Ecoli(C)* and *Heart-Hungarian(M)* with an accuracy boost of 10.04%, 6.15%, 4.81%, 4.72% and 4.65% respectively.

The summary in Table 4.7 makes it quite obvious that the EDISC technique outperforms the other discretisation techniques with respect to *accuracy* and *discretisation time* on a remarkable 71% and 94% of the total number of datasets employed. It also gives fewer rules for 46% of the datasets along with resulting in a minimum execution time for 34% of the datasets. The comparison of all the discretisation techniques in terms of the *number of rules*, *classification accuracy*, *discretisation time* and *execution time* is also represented in graphical form in Figures 4.6 – 4.9 respectively.

4.6 Summary

This chapter has proposed a new discretisation technique EDISC based on a novel concept

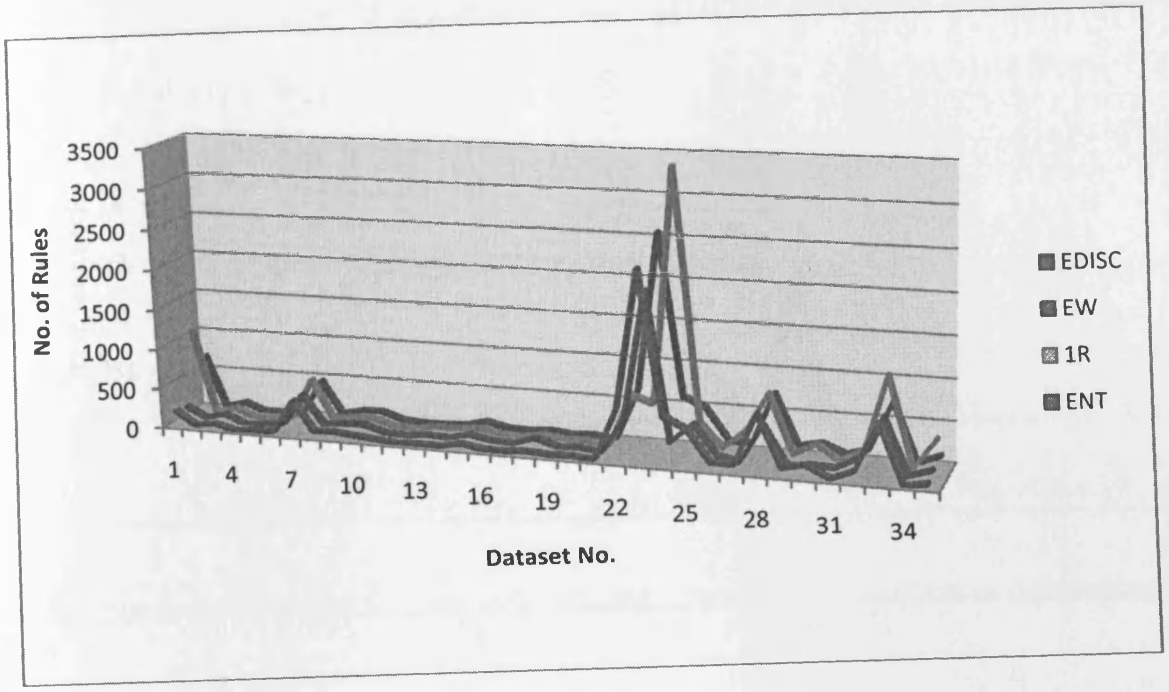


Figure 4.6 Discretisation Techniques Comparison in terms of Number of Rules

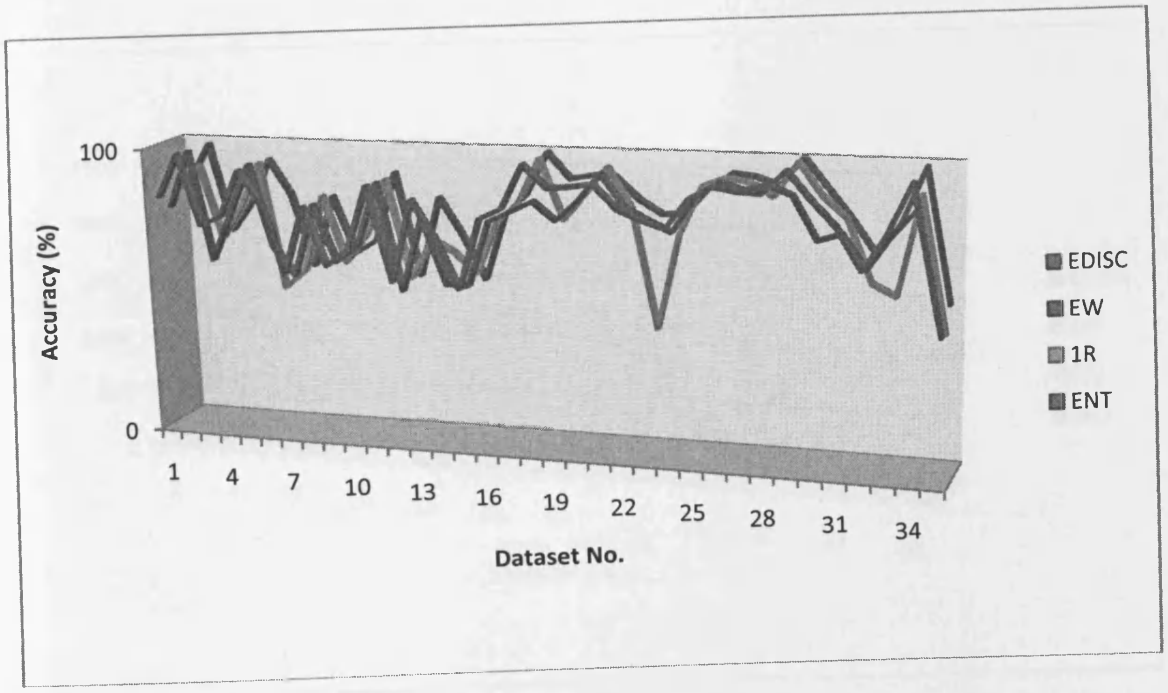


Figure 4.7 Discretisation Techniques Comparison in terms of Accuracy

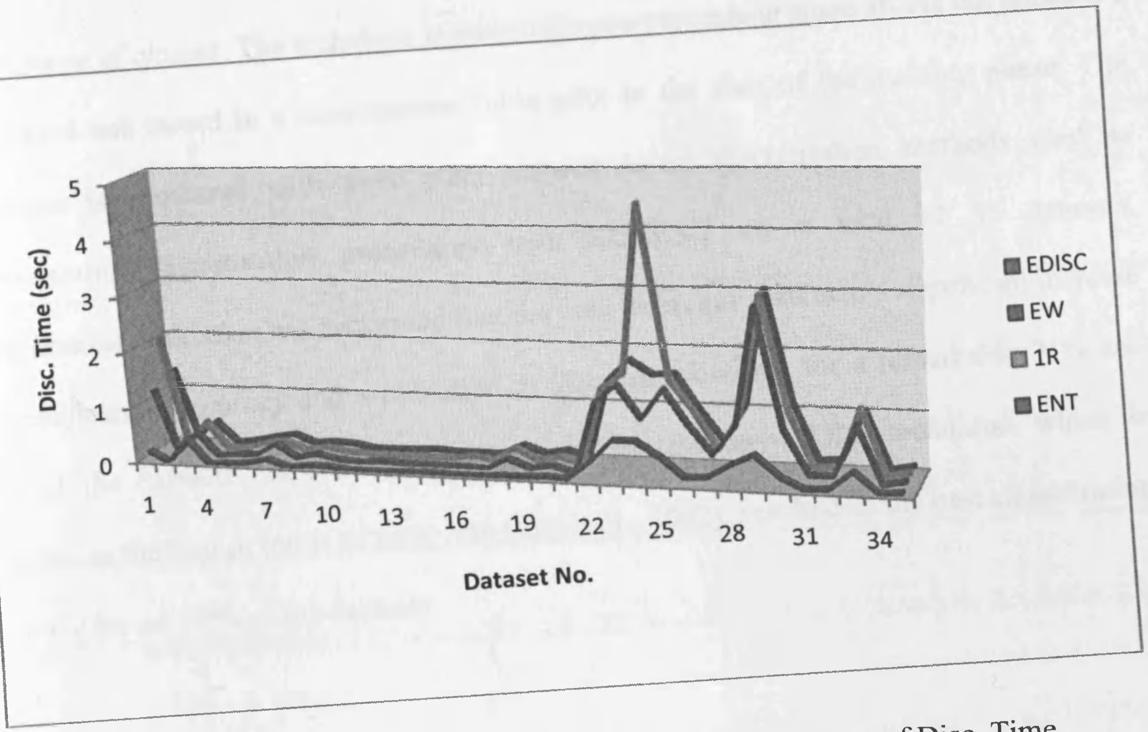


Figure 4.8 Discretisation Techniques Comparison in terms of Disc. Time

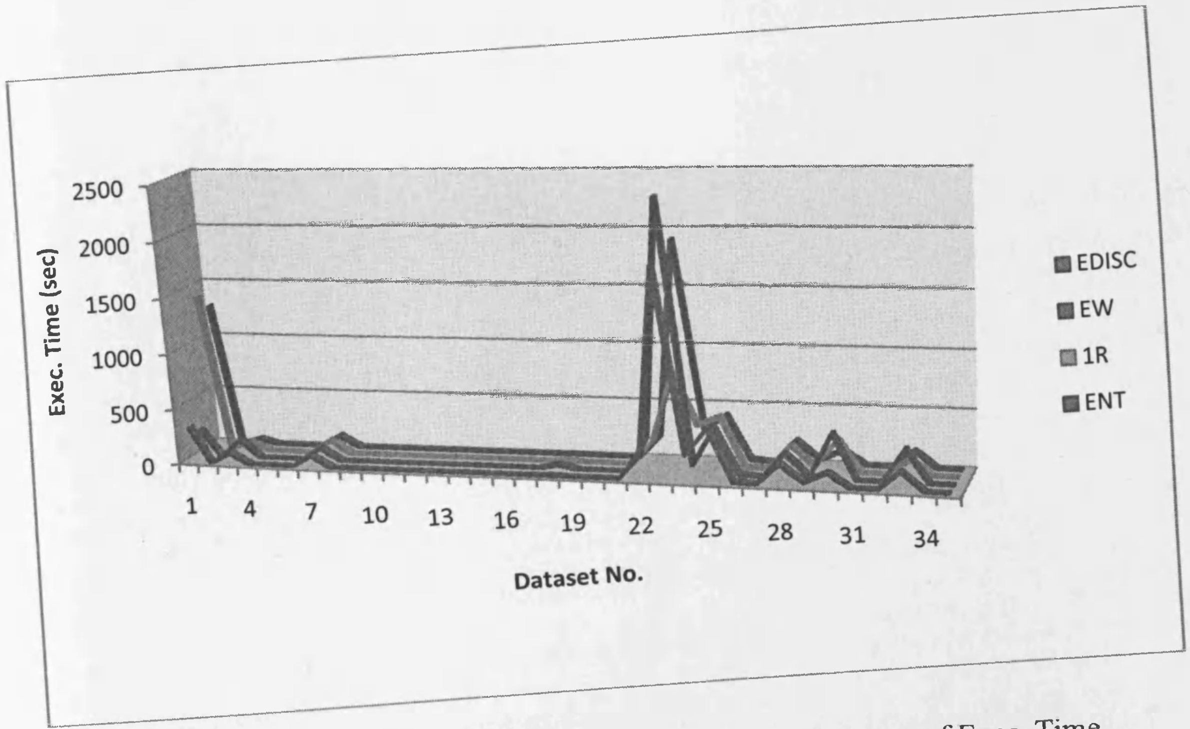


Figure 4.9 Discretisation Techniques Comparison in terms of Exec. Time

called *scope of classes*. The technique is essentially pre-processing since all the cut points are discovered and stored in a discretisation table prior to the start of the learning phase. The technique is compared with three other state-of-the-art discretisation methods used as preprocessing discretisation procedures with RULES-7 on a total of 35 datasets. Experimental evaluation has confirmed that the new technique results in a significant increase in *classification accuracy* and a reduction in *discretisation time* for a remarkable 71% and 94% of the datasets respectively. By contrast, the Entropy-MDLP technique, which is regarded as the best in terms of error rate (Liu et al., 2002), resulted in the best classification accuracy for only 9% of the datasets.

CHAPTER 5

NEW SIMPLE PRUNING TECHNIQUES FOR RULE INDUCTION

5.1 Motivation

Rule induction from large noisy datasets has recently gained importance because of the abundance of such data in the real world. In contrast to artificially generated datasets, real-world datasets almost always contain examples with inconsistencies in either one or more attribute values, the class label or both. Multiple factors may be responsible for contributing to such inconsistencies such as data entry errors, inadequate accuracy of instruments in experimentation, as well as data communication errors (Afify, 2004, Witten and Frank, 2005).

Learning in the presence of noisy data can lead to a phenomenon known as overfitting, in which the learning algorithm tries to model the specific details of the training data to such an extent that the random disturbances in the training set are included in the model as being meaningful (Klawonn and Rehm, 2006). This results in a twofold problem. Firstly, the output of the learning phase is a rule set comprising a large number of rules with low coverage which considerably increases the learning time of the algorithm on the training data. Secondly, the inability of the majority of induced rules to discover any genuine pattern underlying the training data results in poor predictive performance of the induced model on test data.

A practical learning system must be able to handle such large noisy datasets for the above mentioned dual reasons of scalability on the training data as well as accuracy with future unseen data. Pruning is a technique well-recognised within the data mining community for addressing these issues. It emphasises a simplicity-first approach by preferring models with lesser rules and by preferring rules with lesser number of conditions. Consequently, the result of the pruning process is a model that is less accurate with the training data but more accurate with future unseen data (Mingers, 1989). It should however be noted that pruning only minimises overfitting and does not eliminate it altogether.

Three fundamental pruning approaches that have been referred to in the literature include pre-pruning, post-pruning and hybrid pruning, the first two of which were native to decision tree induction algorithms (Breiman et al., 1984, Quinlan, 1983, Quinlan, 1993). However, because of issues such as repetition and replication in decision tree induction (Han and Kamber, 2006) which resulted in poor comprehensibility of the induced models, a different knowledge representation scheme was sought, which led to the emergence of rule induction algorithms (Clark and Niblett, 1989, Michalski, 1969, Quinlan, 1990). Consequently, the first two of the above-mentioned pruning approaches were later adapted for rule induction algorithms (Furnkranz, 1997).

This chapter presents a new hybrid pruning technique and an incremental post-pruning technique for use with RULES-7. The proposed hybrid pruning technique adopts a simple criterion in order to address the problem of overlapping inherent to the RULES family, which results in a more compact rule set with higher classification accuracy. By contrast, the incremental post-pruning technique is based on a misclassification tolerance, which results in minimum execution time as well as a cumulative accuracy comparable to the hybrid one. The latter technique is also applicable to any covering algorithm in general.

This chapter is organised as follows. Section 5.2 reviews some well-known pruning techniques found in the literature. Section 5.3 outlines the proposed pruning techniques for RULES-7. Experimental evaluation of the proposed techniques is presented in section 5.4. Section 5.5 concludes the chapter with a brief summary.

5.2 Survey of Existing Pruning Techniques

Pruning approaches can be broadly classified into three main categories, namely pre-pruning, post-pruning and hybrid pruning. Pre- and post-pruning techniques were originally developed in the context of decision tree induction algorithms. A comprehensive survey as well as a framework for tree-simplification procedures can be found in (Breslow and Aha, 1997), where the first two of the above-mentioned pruning approaches were placed under the broad category of "Controlling Tree Size". As already mentioned in section 5.1, the focus was later shifted to the development of efficient rule induction algorithms, which resulted in a need for adapting these pruning procedures for rule induction as well. A review of several adaptations of pre- and post-pruning techniques for separate-and-conquer rule induction algorithms can be found in (Furnkranz, 1997).

5.2.1 Pre-pruning Techniques

Pre-pruning approaches attempt to handle the noise within the data by preventing the model from overfitting the training data during the learning phase with the help of a variety of stopping criteria. Such a criterion can be employed either during the search for the best rule or after the overall best rule has been returned to the calling procedure. In the former case, its outcome decides whether to specialise the rule further by appending conditions to it. By contrast, in the latter case, the algorithm either continues to induce further rules or stops rule induction based on the outcome of the stopping criterion. In this way, pre-pruning approaches can control the length of a rule, the length of the rule set or both. One of the simplest pre-pruning criteria involves the imposition of a minimum threshold (Li et al., 2001, Liu et al., 1998, Quinlan, 1986, Yin and Han, 2003). Other criteria include the MDL principle (Rissanen, 1986) implemented in FOIL (Quinlan, 1990) and more recently in SRI (Pham and Afify, 2006b), significance testing as used in (Clark and Niblett, 1989) and the cutoff stopping criterion employed in FOSSIL (Furnkranz, 1994a).

5.2.2 Post-pruning Techniques

Post-pruning methods defer the pruning phase until the end of the learning phase as opposed to pre-pruning ones. They first learn a theory that is complete and consistent with the training data and later simplify it. The word "theory" may refer to either an individual rule or a complete rule set. This definition of post-pruning is in agreement with the one in (Furnkranz, 1997), where the algorithm I-REP which uses a pre-pruning stopping criterion and prunes a rule immediately after induction is referred to as one that integrates pre- and post-pruning. It is also confirmed in (Maimon and Rokach, 2007) where post-pruning methods are referred to as those that first produce a complete and consistent rule or rule set, and later try to simplify it. Consequently, this definition of post-pruning will be adopted for the purpose of discussion in the remainder of this chapter.

Post-pruning methods that try to simplify the complete rule set are computationally expensive but are generally viewed as more accurate than pre-pruning ones. One of the earliest post-pruning methods adapted from decision trees for rule induction is REP (Brunk and Pazzani, 1991), which stands for "reduced error pruning". It divides the training set into a growing and a pruning set as per user-specified criterion, learns a complete and consistent theory from the growing set and subsequently prunes it with the objective of amplifying its accuracy on the pruning set. Although accurate, REP has been shown to be highly inefficient because the overfitting theory it starts with after the learning phase can be much more complex than the final pruned theory.

It has also been pointed out that the adaptation of REP for rule induction is incompatible with the covering algorithm approach. This is because the pruning of branches in a decision tree does not affect the neighboring branches. By contrast, the pruning of conditions from a rule generalises the rule due to which its coverage increases. This would result in the removal of a greater number of examples after the induction of that rule, so that the induction of the next rule should be based on a lesser number of examples. Likewise, if the entire first rule is to be pruned, none of examples would be removed and the next rule should be induced based on all the examples. It should be noted however that this issue does not affect the class of algorithms that only mark the covered examples instead of removing them, such as the RULES family. The output of such algorithms is a rule set in which each rule is independent of others and can therefore be pruned without affecting the rest of the rule set.

In order to address the inefficiency issue with REP, a new top-down post-pruning algorithm *Grow* (Cohen, 1993) was proposed by Cohen, based on a technique used by Pagallo and Haussler (Pagallo and Haussler, 1990). The algorithm uses the overfitting theory from the learning phase to grow a pruned theory. It then iteratively selects rules from this grown theory to develop the final theory. However, the algorithm still suffers from the inefficiency caused by the need to generate an overly specific rule set first.

5.2.3 Hybrid-pruning Techniques

Hybrid pruning methods attempt to integrate pre-pruning and post-pruning in order to avoid the expensive learning phase resulting in an overfitting theory. The purpose of the pre-pruning constraints in such a method is to reduce the degree of overfitting so as to increase the efficiency of the subsequent post-pruning phase. Top Down Pruning (TDP) proposed by Furnkranz (Furnkranz, 1994b) is one of the first algorithms exploiting this methodology. The algorithm starts with a pre-pruning phase generating rule sets pruned in a most general to most specific order, followed by a post-pruning phase on the most specific rule set with an accuracy comparable to that of the best rule set up to that point. TDP was empirically shown to be a significant improvement over REP with respect to both runtime and accuracy.

Furnkranz and Widmer later developed another hybrid pruning algorithm I-REP (Furnkranz and Widmer, 1994) which stands for "incremental reduced error pruning". Similar to TDP, I-REP combines pre-pruning and post-pruning so as to avoid learning an overfitting theory. However, in contrast to TDP, as soon as a rule is induced from the growing set, it is pruned based on its accuracy on the pruning set. This post-pruning phase implemented in I-REP has therefore been referred to as incremental post-pruning since it prunes a single rule immediately after induction, contrary to the standard approach which prunes a complete rule set after induction. Because of this, the expensive phase of generating the complete rule set and pruning it afterwards is avoided. Secondly, since the pruned rule is a generalised version of the original rule, the additional examples that it covers are removed before the induction of the next rule, which also solves the incompatibility issue of REP with rule induction.

Furthermore, the pre-pruning stopping criterion in I-REP instructs the algorithm to stop learning as soon as a rule inferior to the most general rule is learned. As a result, the theory learned by I-REP is significantly better than that learned by both REP and *Grow*. However, it has also been shown empirically to perform poorly in domains with a very specific concept description.

All of the above-mentioned post-pruning as well as hybrid pruning methods suffer from a common drawback of having to split the training data into a growing and a pruning set, which results in a multitude of problems. Firstly, the induced rule sets are highly dependent on the adopted splitting methodology. Secondly, the requirement for the pruning set to contain 'at least one example of each disjunctive clause' may not be able to be met in the case of small datasets. Thirdly, the learning phase does not utilise the useful information present in the pruning set for rule induction, resulting in rule sets which are unable to capture the entire concept. Furthermore, it has also been pointed out that the rule sets induced by such pruning techniques may be overfitted to the pruning set (Pham et al., 2004).

In order to address the above-mentioned issues, alternative methods such as (Pfahringer, 1997, Mehta et al., 1995, Pham and Afify, 2006b) based on the MDL principle (Rissanen, 1986) have been proposed. However, these methods also suffer from various drawbacks such as their computationally intensive evaluation approach, their reliance on arbitrary heuristic measures to reach an acceptable level of pruning, and variable performance for different application domains (Pham et al., 2004). The new pruning techniques presented in this chapter solve the above-mentioned problems by being able to operate without a training data split and making the pruning level user-controlled instead of heuristic dependent.

5.3 Proposed Pruning Techniques

This section discusses in detail the two new pruning techniques developed for RULES-7. The first of these can be categorised as a hybrid pruning technique which reduces the total number of rules by addressing the issue of overlapping inherent to the RULES *family* of algorithms. The second one is an incremental post-pruning technique designed specifically to handle the issue of noisy data. These techniques are discussed in detail in sections 5.3.1 and 5.3.2.

5.3.1 Minimum New Classification (MNC)

As already mentioned in section 5.2.2, the RULES family differs from other covering algorithms in that the examples covered by a rule are only marked instead of being removed, which is necessary for the induction process to progress by avoiding the generation of the same rule. Although the phenomenon of overlapping among the rules is common to all covering algorithms, which leads to an increase in the number of rules needed to capture the concept, it is more so in case of algorithms that only mark the examples instead of removing them. This is because all the examples continue to be used for the purpose of calculating both the accuracy as well as the score of each newly formed rule, resulting in a greater degree of overlapping. However, the advantage of taking into account the complete set of examples each time a new rule is formed is that both the *fragmentation* problem (reduction in the amount of data during the later stages of induction) as well as the *small disjuncts* problem (low coverage rules with a high error rate) can be avoided (Afify, 2004).

In order to address the issue of overlapping with the RULES family, a new specialisation heuristic was attempted in (Bigot, 2002) which integrated the information relating to the "New Classified" instances of the rule into the evaluation function. The heuristic is called the "*S measure*" and is given by:

$$S = \frac{p}{p+n} \cdot \frac{p\_new}{P\_unclassified} \cdot \left(1 - \frac{n}{N}\right) \tag{5.1}$$

where p = the number of positive examples covered by new rule

n = the number of negative examples covered by new rule

p_new = the number of previously uncovered positive examples covered by new rule

$P\_unclassified$ = total number of previously uncovered positive examples

N = total number of negative examples

The proposed specialization heuristic reduced the total number of rules by 16.83% while maintaining the classification accuracy for the 15 datasets tested.

As for RULES-6, the specialisation heuristic used in RULES-7 is also the m-probability-estimate (Cestnik, 1990). The approach to integrating the "New Classified" information of the rule (*p_new*) into the specialisation heuristic has also been investigated in this work, but did not yield any significant improvements either. The reason for this is that the integration of *p_new* into the specialisation heuristic does not impose any constraint on the new rule to cover a certain minimum number of previously uncovered positive examples. For instance, if there are two competing *ChildRules* C_1 and C_2 with the same score and with *p_new* equal to 1 and 2 respectively, then the score for C_2 will be slightly higher because of the ratio $p\_new/P\_unclassified$ and so it will be preferred to C_1. However, it is quite obvious that both these rules contribute nothing towards covering a greater number of previously uncovered positive examples.

On the other hand, if the "New Classified" information of the rule is imposed as a constraint, it has the potential to drastically reduce the number of overlapping rules. This is because a rule in that case will be accepted only if it satisfies a certain "Minimum New Classification" criterion (abbreviated MNC) specified by the user. The MNC is defined as the percentage of the total number of instances of the target class which should be covered by the rule from among the instances of that class that have not yet been covered by the rule set created so far.

Using the MNC criterion, a new incremental post-pruning technique was initially attempted in this work which operates as follows. When a rule is returned by the procedure *Induce_One_Rule* to the RULES-7 algorithm at step 6, it is immediately checked for the MNC criterion. If the "New Classified" instances of the rule are less than this criterion, then only the seed example from which the rule is induced is marked as covered and the rule is not added to the rule set. On the other hand, if the number of "New Classified" instances is greater than or equal to MNC and $rule.Classified$ is greater than $rule.Misclassified$, then all examples covered by the rule are marked and the rule is added to the rule set. A pseudo-code description of the RULES-7 algorithm with MNC is outlined in Figure 5.1.

Marking only the seed example in case the rule does not meet the MNC criterion is justifiable since the percentage of newly covered instances of the induced rule is extremely small. Consequently, the rule is deemed to be very similar to the rules already induced and so it is logical to assume that the marked seed example will be covered by one of the rules already induced up to that point by means of approximation. However, the implication of this in the classification phase is that some kind of distance measure must be incorporated to account for the approximation when unseen examples are classified by the rule set formed using MNC. A simple distance measure that can identify the closest rule to a particular test example is given

```
Procedure RULES-7 (Dataset, $w$, $MinSup$, $MinPosSup$, $\lambda$, $MNC\_Th$)
RuleSet = $\emptyset$                                                        (step 1)
Disc_Table = EDISC (DataSet)                                                 (step 2)
Compute the no. of instances a ParentRule should cover as per MinSup         (step 3)
While there are instances in the Dataset that have not been covered Do       (step 4)
  Take a seed example $s$ that has not yet been covered.                     (step 5)
  $MNC = MNC\_Th \times$ No. of Instances of Class of seed $s$               (step 6)
  Rule = Induce_One_Rule ($s$, DataSet, Disc_Table, $w$, MinSup, MinPosSup, $\lambda$)  (step 7)
  If Rule.NewClassified < MNC Then                                           (step 8)
    Mark only the seed example $s$ as covered.                               (step 9)
  Else If Rule.Classified > Rule.Misclassified Then                          (step 10)
    Mark the instances covered by the Rule as covered.                       (step 11)
    Add Rule to the RuleSet                                                  (step 12)
End While
Return RuleSet                                                               (step 13)
End                                                                          (step 14)
```

Figure 5.1 A pseudo-code description of RULES-7 with MNC

by (adapted from (Pham et al., 2003)):

$$Distance = \sqrt{\sum_{d} d\_distance + \sum_{c} c\_distance} \tag{5.2}$$

where

$\sum_d$ = sum for discrete attributes

$d\_distance = 0$ or 1 depending on whether $V_E^i = V_R^i$ or $V_E^i \neq V_R^i$

$\sum_c$ = sum for continuous attributes

$c\_distance \ = 0 \ \text{ if } \ V_{R_{lb}}^i \leq V_E^i \leq V_{R_{ub}}^i$

V_E^i = value of the *i*th attribute in example *E*

$V_{R_{lb}}^i$ = value of lower bound of rule's condition for attribute *i*

$V_{R_{ub}}^i$ = value of upper bound of rule's condition for attribute *i*

$$c\_distance = \left(\frac{V_E^i - V_{R_{lb}}^i}{V_{max}^i - V_{min}^i}\right)^2 \ \text{ if } V_E^i < V_{R_{lb}}^i$$

V_{max}^i, V_{min}^i = minimum and maximum values of the attribute respectively

$$c\_distance = \left(\frac{V_E^i - V_{R_{ub}}^i}{V_{max}^i - V_{min}^i}\right)^2 \ \text{ if } V_E^i > V_{R_{ub}}^i$$

Experimental evaluation has proved that the MNC technique significantly minimises the overlapping among the rules, resulting in a more compact rule set which is also more accurate on test data. The only drawback of the MNC criterion implemented as an incremental post-pruning technique is its high computational requirements. This is because time is wasted in specialising a rule which later has to be pruned because of not covering the required number of "New" instances. Secondly, failure to satisfy the MNC criterion results in only the seed example being marked which also adds to the execution time because a greater number of examples yet remain to be covered.

To address the issue of computational complexity, it is necessary to prune the partial rules created during the search for the best rule in the procedure *Induce_One_Rule*. There are two ways to accomplish this. The first one is to place the MNC criterion as a pre-pruning check at step 5, which will prevent the creation of *ChildRules* of the *ParentRule* in case it does not satisfy the criterion. The second one is to place it in the search space pruning control structure at step 8, which will ensure that the *ChildRule* will replace the last *BestRule* only if it satisfies the MNC criterion, otherwise the last condition used to specialise it will be marked as invalid. This results in a new hybrid pruning technique with two different versions, referred to as MNC^1 and MNC^2.

The modified RULES-7 algorithm and the *Induce_One_Rule* procedure with MNC^1 and MNC^2 are outlined in Figures 5.2 – 5.3 respectively. If MNC^1_Th is set to zero by the user in RULES-7, then MNC^1 will be deactivated at step 6 of *Induce_One_Rule* procedure. Similarly, MNC^2 at step 10 in *Induce_One_Rule* can be deactivated as well by setting its corresponding threshold MNC^2_Th equal to zero in RULES-7. It has been found empirically that the set of optimal values for MNC^1_Th and MNC^2_Th is {1, 5, 10, 15} and {1, 5, 10, 20, 30} respectively. The speed of execution of the algorithm reduces at higher thresholds since a greater number of rules cannot meet the MNC criterion and have to be discarded, which results in more uncovered examples that yet remain to be covered.

```
Procedure RULES-7 (Dataset, $w$, $MinSup$, $MinPosSup$, $\lambda$, $MNC^1\_Th$, $MNC^2\_Th$)
RuleSet = ∅                                                                  (step 1)
$Disc\_Table$ = EDISC (DataSet)                                              (step 2)
Compute the no. of instances a ParentRule should cover as per MinSup        (step 3)
While there are instances in the Dataset that have not been covered Do     (step 4)
  Take a seed example $s$ that has not yet been covered.                     (step 5)
  $MNC^1 = MNC^1\_Th \times$ No. of Instances of Class of seed $s$           (step 6)
  $MNC^2 = MNC^2\_Th \times$ No. of Instances of Class of seed $s$           (step 7)
  If $MNC^1 > 0$ Then                                                        (step 8)
    $MNC = MNC^1$                                                            (step 9)
  Else If $MNC^2 > 0$ Then                                                   (step 10)
    $MNC = MNC^2$                                                            (step 11)
  Rule = Induce_One_Rule ($s$, DataSet, $Disc\_Table$, $w$, $MS$, $MinPosSup$, $\lambda$, $MNC^1$, $MNC^2$)
                                                                             (step 12)
  If Rule.NewClassified < MNC Then                                           (step 13)
    Mark only the seed example $s$ as covered.                               (step 14)
  Else If Rule.Classified > Rule.Misclassified Then                          (step 15)
    Mark the instances covered by the Rule as covered.                       (step 16)
    Add Rule to the RuleSet                                                  (step 17)
End While
Return RuleSet                                                               (step 18)
End                                                                          (step 19)
```

Figure 5.2 A pseudo-code description of RULES-7 with MNC^1 and MNC^2

```
Procedure Induce_One_Rule (s, DataSet, Disc_Table, w, MinSup, MinPosSup, $\lambda$, $MNC^1$,
$MNC^2$)
ParentRuleSet = ChildRuleSet = $\emptyset$
BestRule = Find_Interval (s, Disc_Table)                                    (step 1)
BestRule = rule with no condition (empty antecedent)
Compute the no. of instances a ChildRule should cover within the target class as per
MinPosSup                                                                   (step 2)
Add BestRule to the ParentRuleSet                                           (step 3)
While ParentRuleSet is NOT Empty Do                                         (step 4)
  For each ParentRule $\in$ ParentRuleSet Do
    If ParentRule.Covered $\geq$ MinSupport AND                             (step 5)
        ParentRule.NewClassified $\geq$ $MNC^1$ Then                        (step 6)
      For each nominal Attribute $A_i$ Do
        If Condition$_i$ $\notin$ ParentRule AND Condition$_i$ is not marked as "Invalid" for ParentRule
        Then
          ChildRule = ParentRule $\wedge$ Condition$_i$                     (step 7)
          If ChildRule $\notin$ ChildRuleSet Then                           (step 8)
          {Check the ChildRuleSet for duplicate rule of the candidate ChildRule}
            Compute ChildRule.Instances
            If ChildRule.Classified $\geq$ MinPosSupport AND                (step 9)
                ChildRule.NewClassified $\geq$ $MNC^2$ AND                  (step 10)
              ParentRule.Misclassified – ChildRule.Misclassified $\geq$ MinExcludedNeg Then
                                                                            (step 11)
              Compute ChildRule.Quality
              If (ChildRule.Score > BestRule.Score) OR
                 (ChildRule.Score > $\lambda \times$ BestRule.Score AND
                  ChildRule.NewClassified > BestRule.NewClassified) Then    (step 12)
                 BestRule = ChildRule
              If ChildRule.Consistency $\neq$ 100% Then                     (step 13)
                Add ChildRule to ChildRuleSet                               (step 14)
              Else
                Mark Condition$_i$ as "Invalid" for ParentRule              (step 15)
```

Figure 5.3 A pseudo-code description of the *Induce_One_Rule* procedure of RULES-7 with MNC^1 and MNC^2

```
        Else
          Mark Condition$_i$ as "Invalid" for ParentRule                          (step 16)
      End For
  End For
  Empty ParentRuleSet

  For each ChildRule $\in$ ChildRuleSet Do
    Compute ChildRule.OptimisticScore
    If ChildRule.OptimisticScore $\leq$ BestRule.Score Then                      (step 17)
      Delete the ChildRule from the ChildRuleSet                                 (step 18)
      For each nominal Attribute $A_i$ Do
        If Condition$_i$ $\in$ ChildRule AND Condition$_i$ $\notin$ ParentRule Then      (step 19)
          Mark Condition$_i$ as "Invalid" for ParentRule                          (step 20)
          Exit For Loop
      End For
  End For

  For each ChildRule $\in$ ChildRuleSet Do
  {Mark any Conditions of remaining ChildRules of each ParentRule as Invalid which are
   $\notin$ ParentRule and "marked as Invalid" for the ParentRule}
    For each nominal Attribute $A_i$ Do
      If Condition$_i$ $\in$ ChildRule AND Condition$_i$ $\notin$ ParentRule Then        (step 21)
        If Condition$_i$ is marked as "Invalid" for ParentRule Then               (step 22)
          Mark Condition$_i$ as "Invalid" for ChildRule                           (step 23)
    End For
  End For
  If $w > 1$ Then
    Add $w$ highest Score ChildRules from ChildRuleSet into ParentRuleSet      (step 24)
    Empty ChildRuleSet
End While
Return BestRule
End
```

Figure 5.3 A pseudo-code description of the *Induce_One_Rule* procedure of RULES-7 with MNC^1 and MNC^2 (continued).

5.3.2 Permissible Misclassification (PMC)

This technique is aimed specifically at addressing the issue of noisy data. As already mentioned, this means that a certain percentage of examples exist within the data having one or more errors in either the attribute values or the class label or both. Consequently, some examples appear to be positive (belonging to the target class of the rule) but are actually negative and vice versa.

A common way of handling this issue without requiring a training data split is that the user should have prior knowledge about how much noise is present in the domain at hand. The user inputs this information before the start of the learning phase, which guides the algorithm as to how noise tolerant the induced rules should be. The greater the noise tolerance specified by the user, the lesser the level of consistency of the induced rules will be. Such a tolerance in rule consistency is not only acceptable but also desirable in the case of noisy data (Pham and Afify, 2006a).

The proposed incremental post-pruning technique works in a fashion similar to I-REP in that an attempt is made to generalise the rule immediately after induction. However, it is different from I-REP in several ways. Firstly, the training data is not split into a growing and a pruning set. Instead, the user specifies a certain permissible misclassification level using domain knowledge in the form of a threshold referred to as "PMC". Although counter examples may also exist in the target class of the rule, it is not so straightforward to take these into account when forming noise tolerant rules, since the goal in rule induction tasks is to maximise the coverage of the positive class. As a result, it is not appropriate to state that a certain number of positive examples should remain uncovered. Accordingly, PMC is defined as the percentage of the total number of negative examples in the dataset (not belonging to the target class of the rule) that the rule should cover in order to account for the noise. If no information is available concerning the noise, the user can still arrive at the best possible rule set by inputting a value out of {1, 5, 10, 20, 30}, which is the set of optimal values found empirically for this parameter.

Another major difference from I-REP lies in the evaluation of conditions when a rule is being considered for pruning. The I-REP technique continues to drop conditions from a rule greedily until its accuracy on the pruning set decreases. The proposed technique, on the other hand, calculates the ratio $rule.Misclassified/rule.Covered$ for each rule resulting from dropping a single condition from the original induced rule and retains the rule for which this ratio is a minimum. If the misclassified instances of the resulting rule are still less than the user-specified PMC, it is further generalised as mentioned above and this process continues as long as the misclassified instances of the resulting rule remain less than PMC. When $rule.Misclassified$ exceeds the PMC level, pruning stops and the last rule for which the number of misclassified instances was less than PMC is chosen as the final rule.

A rule is considered for generalisation only if $rule.Classified$ is greater than $rule.Misclassified$, in which case the overall concept accuracy may increase, otherwise it is pruned right away. If this criterion is satisfied by the rule, it passes through yet another check which determines whether $rule.Covered$ is less than 10% of the total number of examples in the dataset. If this is the case, then the rule is generalised, otherwise it is added to the rule set. The rationale for this is straightforward. Rules with a lesser coverage level are more prone to overfitting the noise than those with a sufficiently high coverage. The modified RULES-7 algorithm after incorporating the PMC technique is outlined in Figure 5.4, while the *Induce_One_Rule* procedure remains the same as in Figure 5.3. The *Generalise_Rule*

```
Procedure RULES-7 (Dataset, $w$, $MinSup$, $MinPosSup$, $\lambda$, $MNC^1\_Th$, $MNC^2\_Th$, $PMC\_Th$,
$Prune\_Method$)
RuleSet = $\emptyset$                                                              (step 1)
$Disc\_Table$ = EDISC (DataSet)                                                    (step 2)
Compute the no. of instances a $ParentRule$ should cover as per $MinSup$           (step 3)
While there are instances in the Dataset that have not been covered Do             (step 4)
  Take a seed example $s$ that has not yet been covered.                           (step 5)
  $MNC^1 = MNC^1\_Th \times$ No. of Instances of Class of seed $s$                 (step 6)
  $MNC^2 = MNC^2\_Th \times$ No. of Instances of Class of seed $s$                 (step 7)
  $PMC = PMC\_Th \times$ (Total No. of Instances in Dataset $-$ No. of Instances
                          of Class of seed $s$)                                    (step 8)
  If $MNC^1 > 0$ Then                                                              (step 9)
    $MNC = MNC^1$                                                                  (step 10)
  Else If $MNC^2 > 0$ Then                                                         (step 11)
    $MNC = MNC^2$                                                                  (step 12)
  $Rule$ = Induce_One_Rule ($s$, DataSet, $Disc\_Table$, $w$, $MS$, $MinPosSup$, $\lambda$, $MNC^1$, $MNC^2$)
                                                                                   (step 13)
  If $Prune\_Method$ = MNC AND $Rule$.NewClassified < MNC Then                     (step 14)
    Mark only the seed example $s$ as covered.                                     (step 15)
  Else If $Rule$.Classified > $Rule$.Misclassified Then                            (step 16)
    $FinalRule = Rule$                                                             (step 17)
    If $Prune\_Method$ = PMC AND $Rule$.Covered < (0.1 $\times$ Total No. of Instances
                                               in Dataset) Then                    (step 18)
      While $Rule$.Misclassified < $PMC$ Do                                        (step 19)
        $FinalRule = Rule$                                                         (step 20)
        $Rule$ = Generalize_Rule ($Rule$, Dataset)                                 (step 21)
      End While
    Mark the instances covered by the $FinalRule$ as covered.                      (step 22)
    Add $FinalRule$ to the RuleSet                                                 (step 23)
End While
Return RuleSet                                                                     (step 24)
End                                                                                (step 25)
```

Figure 5.4 Modified pseudo-code description of RULES-7 with PMC

procedure at step 21 of RULES-7 is outlined in Figure 5.5. The *Prune_Method* selected by the user decides which of the two steps 14 or 18 is deactivated in RULES-7.

5.4 Empirical Evaluation of Proposed Pruning Techniques

The pruning techniques suggested in this chapter have been tested thoroughly on a population of 40 datasets, all of which have been downloaded from the repository of machine learning databases, University of California at Irvine (UCI) (Blake and Merz, 1998), with the exception of the dataset *Depression* from Williams College (Veaux, 2007). These datasets come from a variety of domains and have been summarised in Table 5.1. Each dataset name is followed by a block capital letter in parenthesis which indicates whether the dataset is nominal, continuous or mixed type. Occasionally, there might also be a suffix L to denote that the dataset is large. The only sampled dataset used in this study is *Cover-Type(M)L* with 581,012 instances which was reduced to approx. 1/187 of its full size.

All evaluation was carried out on the Windows XP operating system using an Intel Pentium 2.0 GHz Dual-Core computer with 2 GB of RAM. In order to handle continuous and mixed type datasets, RULES-7 uses its own EDISC discretisation technique, which discovers the class-specific cut points for each attribute prior to the learning phase. The evaluation approach used in this study is the standard *stratified 10-fold cross-validation* (Kohavi, 1995a) so as to ensure that the prediction estimate is as accurate as possible. Three commonly used criteria, namely *Rules_Reduction*, *Accuracy_Increase* and *Time_Reduction* have been used for testing the proposed pruning techniques. Additionally, an *Overall Improvement Criterion* has also been defined, which is satisfied by a dataset if any one of the above-mentioned three criteria is equal to zero and the other two are greater than zero. All execution times have been reported in seconds.

```
Procedure Generalise_Rule (Rule, Dataset)
Min_Ratio = MAX                                                    (step 1)
For each nominal Attribute $A_i$ Do                                (step 2)
  If Condition$_i$ $\in$ Rule                                      (step 3)
    Mark Condition$_i$ as "Not Existing" for Rule                  (step 4)
    Compute Rule.Instances
    If $Rule.Misclassified/Rule.Covered$ < Min_Ratio Then          (step 5)
      Min_Ratio = $Rule.Misclassified/Rule.Covered$                (step 6)
      Condition$_{MR}$ = Condition$_i$                             (step 7)
    Mark Condition$_i$ as "Existing" for Rule                      (step 8)
Mark Condition$_{MR}$ as "Not Existing" for Rule                   (step 9)
Return Rule                                                        (step 10)
```

Figure 5.5 A pseudo-code description of the *Generalise_Rule* procedure

Condition$_{MR}$ = Condition whose removal yields Min_Ratio

No.	Dataset	No. of Instances	No. of Attributes			No. of Classes (k)
			Total	Nominal	Continuous	
1	Arrhythmia(M)L	452	279	73	206	13
2	Balance-Scale(N)	625	4	4	0	3
3	Breast-Cancer(C)	699	10	0	10	2
4	Breast-Cancer(N)	286	9	9	0	2
5	Car(N)	1,728	6	6	0	4
6	Chess(N)L	3,196	36	36	0	2
7	Connectionist-Sonar(C)	208	60	0	60	2
8	*Cover-Type(M)L	3,100	54	47	7	7
9	Credit-Approval(M)	690	15	9	6	2
10	Cylinder-Bands(M)	541	39	19	20	2
11	Depression(M)	428	17	11	6	2
12	Ecoli(C)	336	8	0	8	8
13	Flags(M)	194	29	29	0	8
14	German-Credit(M)	1,000	20	13	7	2
15	Glass(C)	214	10	0	10	6
16	Heart-Cleveland(M)	303	13	8	5	5
17	Heart-Hungarian(M)	294	13	8	5	2
18	Hepatitis(M)	155	19	13	6	2
19	Horse-Colic(M)	368	27	18	9	2
20	Hyperthyroid(M)L	3,711	29	22	7	4
21	Hypothyroid(M)L	3,772	29	22	7	2
22	Image(C)	210	19	0	19	7
23	Ionosphere(C)	351	34	0	34	2
24	Iris(C)	150	4	0	4	3
25	Landsat(C)L	6,435	36	0	36	6
26	Mushroom(N)L	8,124	22	22	0	2
27	Nursery(N)L	12,960	8	8	0	2
28	Page-Blocks(C)L	5,473	10	0	10	5
29	Parkinsons(C)	195	22	22	0	2
30	P-O-Patient(M)	90	8	8	0	3
31	Promoters(N)	106	58	58	0	2
32	Shuttle(C)L	58,000	5	0	5	7
33	Soybean-Large(N)	683	35	35	0	19
34	Spambase(C)L	4,601	57	0	57	2
35	Spect(C)	267	44	0	44	2
36	SPECT-Heart(N)	267	22	22	0	2
37	Splice(N)L	3,190	61	61	0	3
38	Tic-Tac-Toe(N)	958	9	9	0	2
39	Vehicle(C)	846	18	0	18	4
40	Waveform-v2(C)L	5,000	40	0	40	3

Table 5.1 Summary of Datasets used in Experiments

Section 5.4.1 compares RULES-7 with MNC[1] and MNC[2] against RULES-7 without any pruning technique activated. The parameters *beam width*, *MinPositives* and *MinExcludedNeg* were set to 4, 2 and 1 respectively. Section 5.4.2 discusses the results obtained with RULES-7 by incorporating the PMC technique against those of the default RULES-7 algorithm. Finally, section 5.4.3 summarises the results presented in sections 5.4.1 – 5.4.2.

5.4.1 RULES-7@MNC vs. RULES-7

Table 5.2 and Table 5.4 present the results obtained from RULES-7 by activating MNC[1] and MNC[2] respectively, and Table 5.3 and Table 5.5 summarise the results for these two versions of MNC as compared with the default RULES-7. A graphical representation of Table 5.3 and Table 5.5 is also presented in Figures 5.6 – 5.8 and Figures 5.9 – 5.11 respectively. It can be seen from the last row in Table 5.2 that MNC[1] reduces the cumulative number of rules by 73.66 % along with increasing the classification accuracy by 2.76%. However, the cumulative execution time is more than double as compared with the default RULES-7. Even so, it is less than the incremental post-pruning technique MNC initially proposed in section 5.3.1, for which the execution time almost tripled. The MNC[2] version solves the inefficiency problem by reducing the cumulative execution time to 30.33%, as indicated by the last row in Table 5.4. Furthermore, it also appears to be superior from the point of view of aggregate rules reduction and accuracy increase, the values for which are 75.12% and 3.61% respectively. However, in terms of the total number of datasets for which the model obtained by pruning resulted in maximum classification accuracy, there is a close tie between the two versions as indicated by Table 5.3 and Table 5.5.

No.	Dataset	MNC[1] ∈ {1, 5, 10, 15} %	No. of Rules		Redc. (%)	Accuracy (%)		Incr. (%)	Exec. Time		Redc. (%)	Speed Boost (x)	Impr. ?
			Rules7	Rules7 @MNC[1]		Rules7	Rules7 @MNC[1]		Rules7	Rules7 @MNC[1]			
1	Arrythmia(M)L	1	71	57	**19.72**	74.00	76.25	**3.04**	93.71	145.35	**-55.11**	**nil**	**no**
2	Balance-Scale(N)	5	113	17	**84.96**	58.83	75.17	**27.76**	0.14	0.17	**-21.43**	**nil**	**no**
3	Breast-Cancer(C)	15	25	3	**88.00**	93.48	95.22	**1.86**	0.11	0.04	**63.64**	**2.8x**	**yes**
4	Breast-Cancer(N)	10	74	5	**93.24**	66.07	71.79	**8.65**	0.22	0.24	**-9.09**	**nil**	**no**
5	Car(N)	15	94	4	**95.74**	61.35	74.97	**22.21**	0.75	0.83	**-10.67**	**nil**	**no**
6	Chess(N)L	1	48	21	**56.25**	94.21	96.86	**2.80**	19.30	26.17	**-35.60**	**nil**	**no**
7	Connectionist(C)	1	28	28	**0.00**	83.00	83.00	**0.00**	0.65	0.67	**-3.08**	**nil**	**no**
8	*Cover-Type(M)L	5	446	32	**92.83**	58.34	61.89	**6.09**	134.96	576.68	**-327.30**	**nil**	**no**
9	Credit-Approval(M)	15	77	5	**93.51**	81.91	85.44	**4.31**	1.40	1.57	**-12.14**	**nil**	**no**
10	Cylinder-Bands(M)	1	88	48	**45.45**	61.89	65.85	**6.40**	3.40	3.50	**-2.94**	**nil**	**no**
11	Depression(M)	10	58	4	**93.10**	67.86	70.95	**4.56**	0.75	0.40	**46.67**	**1.9x**	**yes**
12	Ecoli(C)	1	23	19	**17.39**	85.81	86.45	**0.75**	0.05	0.06	**-20.00**	**nil**	**no**
13	Flags(M)	15	52	17	**67.31**	61.88	67.50	**9.09**	0.43	0.26	**39.53**	**1.7x**	**yes**
14	German-Credit(M)	5	240	8	**96.67**	71.00	72.10	**1.55**	6.45	9.85	**-52.71**	**nil**	**no**
15	Glass(C)	1	18	20	**-11.11**	72.78	74.44	**2.29**	0.05	0.08	**-60.00**	**nil**	**no**
16	Heart-Cleveland(M)	1	69	67	**2.90**	56.07	58.57	**4.46**	0.45	0.58	**-28.89**	**nil**	**no**
17	Heart-Hungarian(M)	1	25	22	**12.00**	80.36	82.14	**2.22**	0.12	0.12	**0.00**	**nil**	**yes**
18	Hepatitis(M)	1	24	18	**25.00**	84.00	84.00	**0.00**	0.10	0.10	**0.00**	**nil**	**no**
19	Horse-Colic(M)	15	49	4	**91.84**	76.39	83.06	**8.73**	0.51	0.60	**-17.65**	**nil**	**no**
20	Hyperthyroid(M)L	5	48	12	**75.00**	98.35	98.48	**0.14**	11.51	21.13	**-83.58**	**nil**	**no**
21	Hypothyroid(M)L	15	41	8	**80.49**	99.02	98.30	**-0.73**	9.70	14.49	**-49.38**	**nil**	**no**
22	Image(C)	5	24	18	**25.00**	91.43	93.33	**2.08**	0.11	0.09	**18.18**	**1.2x**	**yes**
23	Ionosphere(C)	1	33	20	**39.39**	92.06	92.35	**0.32**	0.41	0.36	**12.20**	**1.1x**	**yes**
24	Iris(C)	10	6	4	**33.33**	93.33	96.00	**2.86**	0.00	0.00	**0.00**	**nil**	**yes**
25	Landsat(C)L	1	647	131	**79.75**	85.36	84.28	**-1.26**	233.09	425.51	**-82.55**	**nil**	**no**
26	Mushroom(N)L	1	26	15	**42.31**	99.56	97.51	**-2.06**	8.01	6.55	**18.23**	**1.2x**	**no**
27	Nursery(N)L	1	357	81	**77.31**	65.73	68.76	**4.61**	76.80	0.00	**100.00**	**max**	**yes**
28	Page-Blocks(C)L	1	88	26	**70.45**	92.50	90.86	**-1.77**	4.69	5.99	**-27.72**	**nil**	**no**
29	Parkinsons(C)	1	21	17	**19.05**	84.44	83.89	**-0.66**	0.15	0.15	**0.00**	**nil**	**no**
30	P-O-Patient(M)	10	28	3	**89.29**	67.50	72.50	**7.41**	0.03	0.03	**0.00**	**nil**	**yes**
31	Promoters(N)	1	20	20	**0.00**	84.00	84.00	**0.00**	0.17	0.18	**-5.88**	**nil**	**no**
32	Shuttle(C)L	1	99	48	**51.52**	99.83	99.01	**-0.81**	44.90	140.18	**-212.20**	**nil**	**no**
33	Soybean-Large(N)	5	54	35	**35.19**	90.16	90.94	**0.87**	1.79	1.56	**12.85**	**1.1x**	**yes**
34	Spambase(C)L	1	157	43	**72.61**	90.54	85.71	**-5.33**	121.11	238.40	**-96.85**	**nil**	**no**
35	Spect(C)	15	15	6	**60.00**	82.31	83.85	**1.87**	0.65	0.54	**16.92**	**1.2x**	**yes**
36	SPECT-Heart(N)	15	34	3	**91.18**	82.69	87.31	**5.58**	0.45	0.42	**6.67**	**1.1x**	**yes**
37	Splice(N)L	1	254	77	**69.69**	90.35	90.00	**-0.39**	253.92	226.64	**10.74**	**1.1x**	**no**
38	Tic-Tac-Toe(N)	1	27	21	**22.22**	92.74	97.16	**4.77**	0.23	0.20	**13.04**	**1.2x**	**yes**
39	Vehicle(C)	1	130	81	**37.69**	68.78	69.76	**1.42**	3.46	4.34	**-25.43**	**nil**	**no**
40	Waveform-v2(C)L	1	718	104	**85.52**	80.89	79.97	**-1.14**	183.60	775.78	**-322.54**	**nil**	**no**
Average:			111	29	**73.66**	80.52	82.74	**2.76**	30.46	65.75	**-115.85**		

Table 5.2 RULES-7@MNC^1 vs. RULES-7

No.	Criterion	No. of Datasets for which					
		Redc.		Equal		Incr.	
		Total	%age	Total	%age	Total	%age
1	No. of Rules	37	**93%**	2	**4%**	1	**3%**
2	Accuracy	9	**23%**	3	**7%**	28	**70%**
3	Exec. Time	12	**30%**	5	**13%**	23	**57%**
4	Overall Impr.	13					**33%**

Table 5.3 Summary of RULES-7@MNC[1]

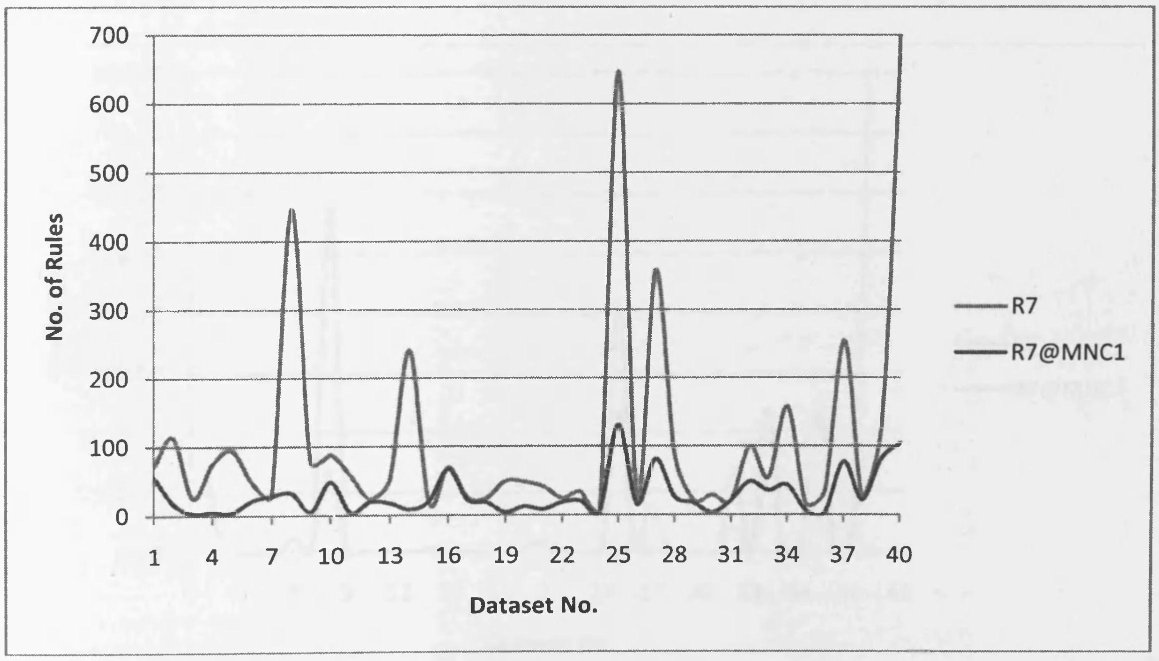


Figure 5.6 RULES-7@MNC[1] vs. RULES-7 in terms of Number of Rules

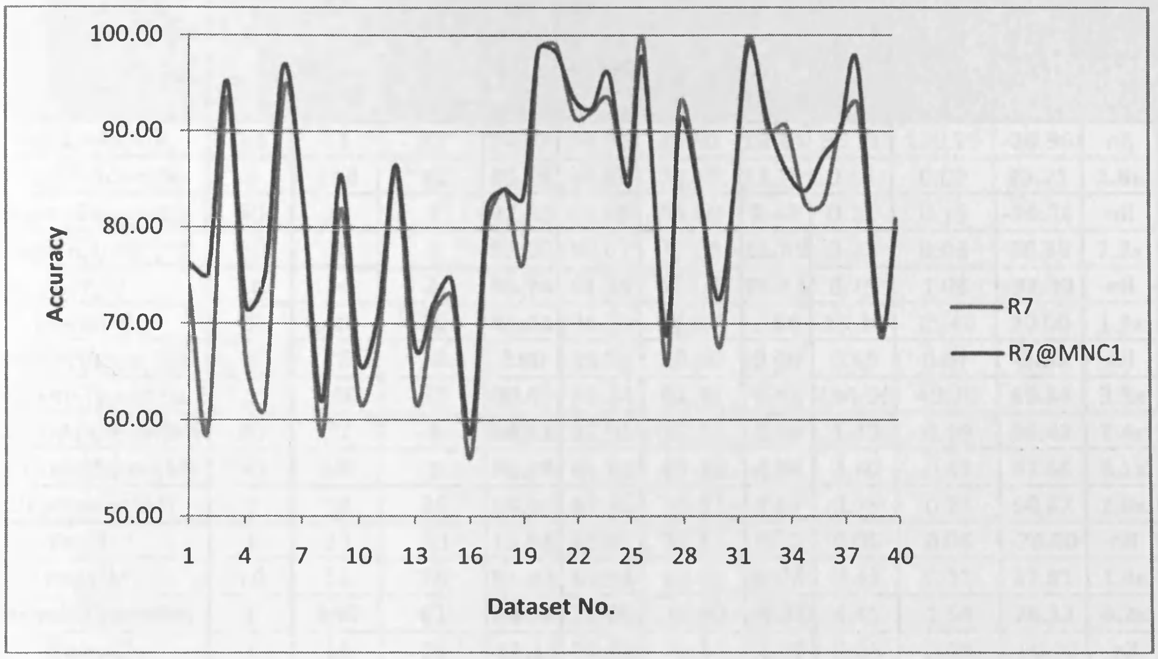


Figure 5.7 RULES-7@MNC^1 vs. RULES-7 in terms of Accuracy

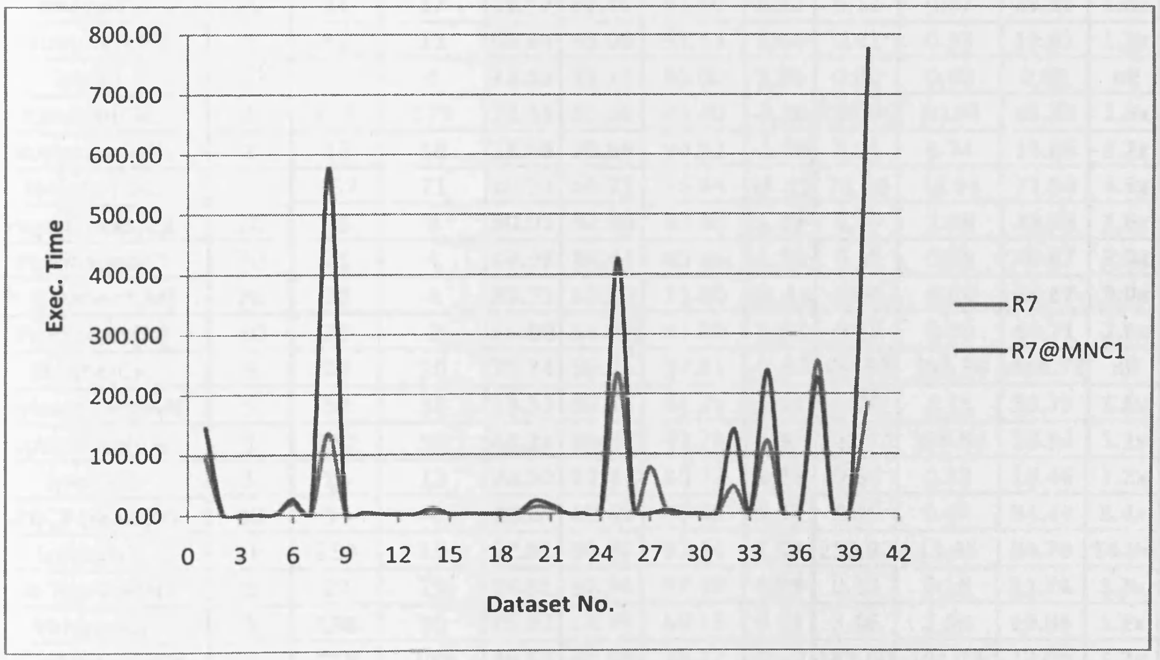


Figure 5.8 RULES-7@MNC^1 vs. RULES-7 in terms of Exec. Time

No.	Dataset	MNC^2 ∈ {1, 5, 10, 20, 30} %	No. of Rules		Redc. (%)	Accuracy (%)		Incr. (%)	Exec. Time		Redc. (%)	Speed Boost (x)	Impr. ?
			Rules7	Rules7 @MNC^2		Rules7	Rules7 @MNC^2		Rules7	Rules7 @MNC^2			
1	Arrythmia(M)L	10	71	32	**54.93**	74.00	81.50	**10.14**	93.71	120.29	**-28.36**	**nil**	no
2	Balance-Scale(N)	5	113	19	**83.19**	58.83	71.33	**21.25**	0.14	0.09	**35.71**	**1.6x**	yes
3	Breast-Cancer(C)	30	25	2	**92.00**	93.48	95.80	**2.48**	0.11	0.15	**-36.36**	**nil**	no
4	Breast-Cancer(N)	20	74	2	**97.30**	66.07	73.93	**11.89**	0.22	0.03	**86.36**	**7.3x**	yes
5	Car(N)	20	94	4	**95.74**	61.35	75.03	**22.31**	0.75	1.03	**-37.33**	**nil**	no
6	Chess(N)L	1	48	26	**45.83**	94.21	96.67	**2.60**	19.30	15.44	**20.00**	**1.3x**	yes
7	Connectionist (C)	1	28	28	**0.00**	83.00	83.00	**0.00**	0.65	0.67	**-3.08**	**nil**	no
8	*Cover-Type(M)L	1	446	87	**80.49**	58.34	62.41	**6.98**	134.96	40.70	**69.84**	**3.3x**	yes
9	Credit-Approval(M)	30	77	4	**94.81**	81.91	82.21	**0.36**	1.40	0.19	**86.43**	**7.4x**	yes
10	Cylinder-Bands(M)	30	88	3	**96.59**	61.89	67.36	**8.84**	3.40	0.42	**87.65**	**8.1x**	yes
11	Depression(M)	1	58	25	**56.90**	67.86	73.33	**8.07**	0.75	0.37	**50.67**	**2.0x**	yes
12	Ecoli(C)	1	23	20	**13.04**	85.81	86.45	**0.75**	0.05	0.06	**-20.00**	**nil**	no
13	Flags(M)	10	52	25	**51.92**	61.88	66.88	**8.08**	0.43	0.31	**27.91**	**1.4x**	yes
14	German-Credit(M)	1	240	61	**74.58**	71.00	70.80	**-0.28**	6.45	1.54	**76.12**	**4.2x**	no
15	Glass(C)	1	18	20	**-11.11**	72.78	74.44	**2.29**	0.05	0.08	**-60.00**	**nil**	no
16	Heart-Cleveland(M)	10	69	14	**79.71**	56.07	61.07	**8.92**	0.45	0.35	**22.22**	**1.3x**	yes
17	Heart-Hungarian(M)	30	25	4	**84.00**	80.36	83.57	**4.00**	0.12	0.00	**100.00**	**max**	yes
18	Hepatitis(M)	5	24	13	**45.83**	84.00	84.00	**0.00**	0.10	0.05	**50.00**	**2.0x**	yes
19	Horse-Colic(M)	1	49	2	**95.92**	76.39	79.44	**4.00**	0.51	0.20	**60.78**	**2.6x**	yes
20	Hyperthyroid(M)L	1	48	26	**45.83**	98.35	98.10	**-0.25**	11.51	10.01	**13.03**	**1.1x**	no
21	Hypothyroid(M)L	1	41	25	**39.02**	99.02	98.51	**-0.51**	9.70	9.08	**6.39**	**1.1x**	no
22	Image(C)	10	24	17	**29.17**	91.43	91.90	**0.52**	0.11	0.07	**36.36**	**1.6x**	yes
23	Ionosphere(C)	5	33	12	**63.64**	92.06	93.53	**1.60**	0.41	0.33	**19.51**	**1.2x**	yes
24	Iris(C)	10	6	4	**33.33**	93.33	96.00	**2.86**	0.00	0.00	**0.00**	**nil**	yes
25	Landsat(C)L	1	647	179	**72.33**	85.36	83.40	**-2.30**	233.09	80.84	**65.32**	**2.9x**	no
26	Mushroom(N)L	1	26	16	**38.46**	99.56	97.82	**-1.75**	8.01	6.74	**15.86**	**1.2x**	no
27	Nursery(N)L	1	357	71	**80.11**	65.73	75.84	**15.39**	76.80	16.94	**77.94**	**4.5x**	yes
28	Page-Blocks(C)L	20	88	8	**90.91**	92.50	93.60	**1.19**	4.69	2.85	**39.23**	**1.6x**	yes
29	Parkinsons(C)	30	21	4	**80.95**	84.44	85.56	**1.32**	0.15	0.05	**66.67**	**3.0x**	yes
30	P-O-Patient(M)	20	28	4	**85.71**	67.50	75.00	**11.11**	0.03	0.01	**66.67**	**3.0x**	yes
31	Promoters(N)	10	20	9	**55.00**	84.00	87.00	**3.57**	0.17	0.06	**64.71**	**2.8x**	yes
32	Shuttle(C)L	5	99	26	**73.74**	99.83	97.81	**-2.02**	44.90	264.36	**-488.78**	**nil**	no
33	Soybean-Large(N)	5	54	36	**33.33**	90.16	91.25	**1.21**	1.79	1.15	**35.75**	**1.6x**	yes
34	Spambase(C)L	1	157	53	**66.24**	90.54	92.28	**1.92**	121.11	105.56	**12.84**	**1.1x**	yes
35	Spect(C)	5	15	12	**20.00**	82.31	85.77	**4.21**	0.65	0.53	**18.46**	**1.2x**	yes
36	SPECT-Heart(N)	30	34	4	**88.24**	82.69	86.92	**5.12**	0.45	0.07	**84.44**	**6.4x**	yes
37	Splice(N)L	10	254	18	**92.91**	90.35	92.64	**2.53**	253.92	13.45	**94.70**	**18.9x**	yes
38	Tic-Tac-Toe(N)	1	27	23	**14.81**	92.74	97.37	**4.99**	0.23	0.18	**21.74**	**1.3x**	yes
39	Vehicle(C)	5	130	30	**76.92**	68.78	69.15	**0.53**	3.46	2.80	**19.08**	**1.2x**	yes
40	Waveform-v2(C)L	1	718	139	**80.64**	80.89	78.37	**-3.12**	183.60	151.73	**17.36**	**1.2x**	no
Average:			111	28	**75.12**	80.52	83.43	**3.61**	30.46	21.22	**30.33**		

Table 5.4 RULES-7@MNC^2 vs. RULES-7

<table>
<tr><th rowspan="3">No.</th><th rowspan="3">Criterion</th><th colspan="6">No. of Datasets for which</th></tr>
<tr><th colspan="2">Redc.</th><th colspan="2">Equal</th><th colspan="2">Incr.</th></tr>
<tr><th>Total</th><th>%age</th><th>Total</th><th>%age</th><th>Total</th><th>%age</th></tr>
<tr><td>1</td><td>No. of Rules</td><td>38</td><td>95%</td><td>1</td><td>3%</td><td>1</td><td>2%</td></tr>
<tr><td>2</td><td>Accuracy</td><td>7</td><td>18%</td><td>1</td><td>2%</td><td>32</td><td>80%</td></tr>
<tr><td>3</td><td>Exec. Time</td><td>32</td><td>80%</td><td>1</td><td>2%</td><td>7</td><td>18%</td></tr>
<tr><td>4</td><td>Overall Impr.</td><td colspan="5">27</td><td>68%</td></tr>
</table>

Table 5.5 Summary of RULES-7@MNC^2

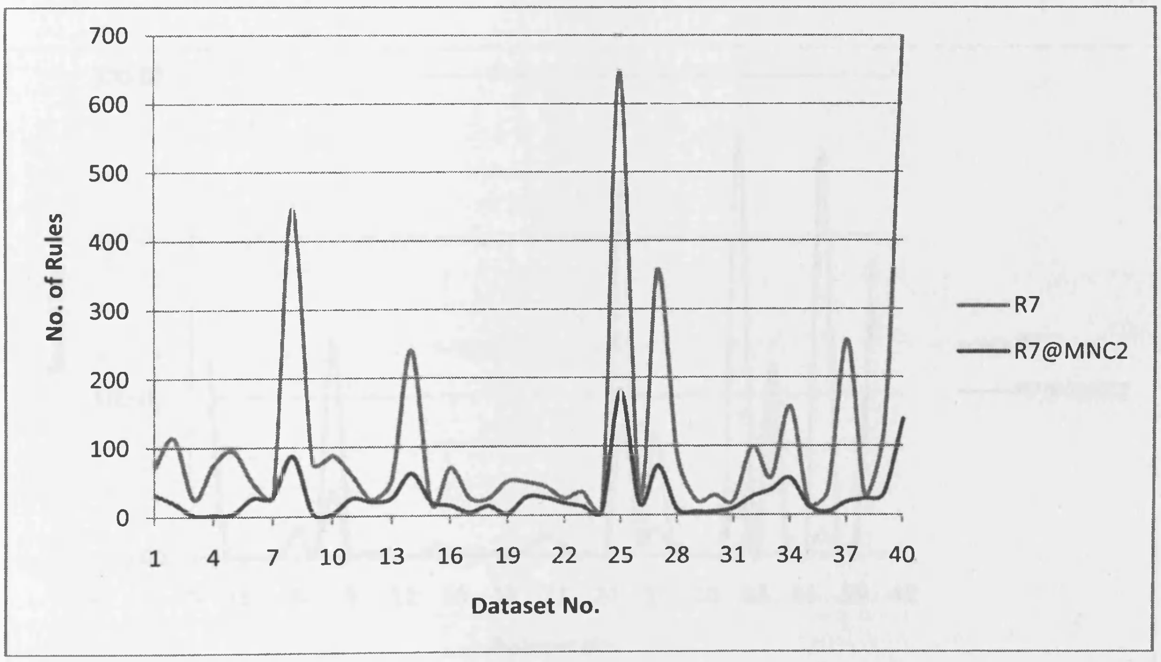


Figure 5.9 RULES-7@MNC^2 vs. RULES-7 in terms of Number of Rules

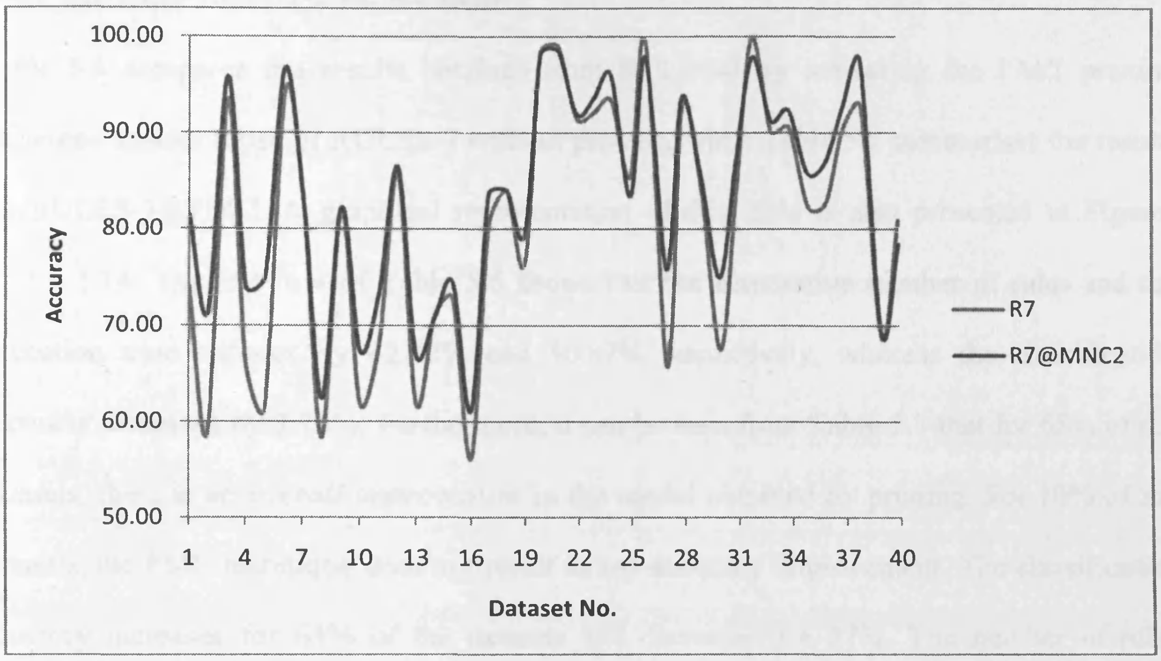


Figure 5.10 RULES-7@MNC^2 vs. RULES-7 in terms of Accuracy

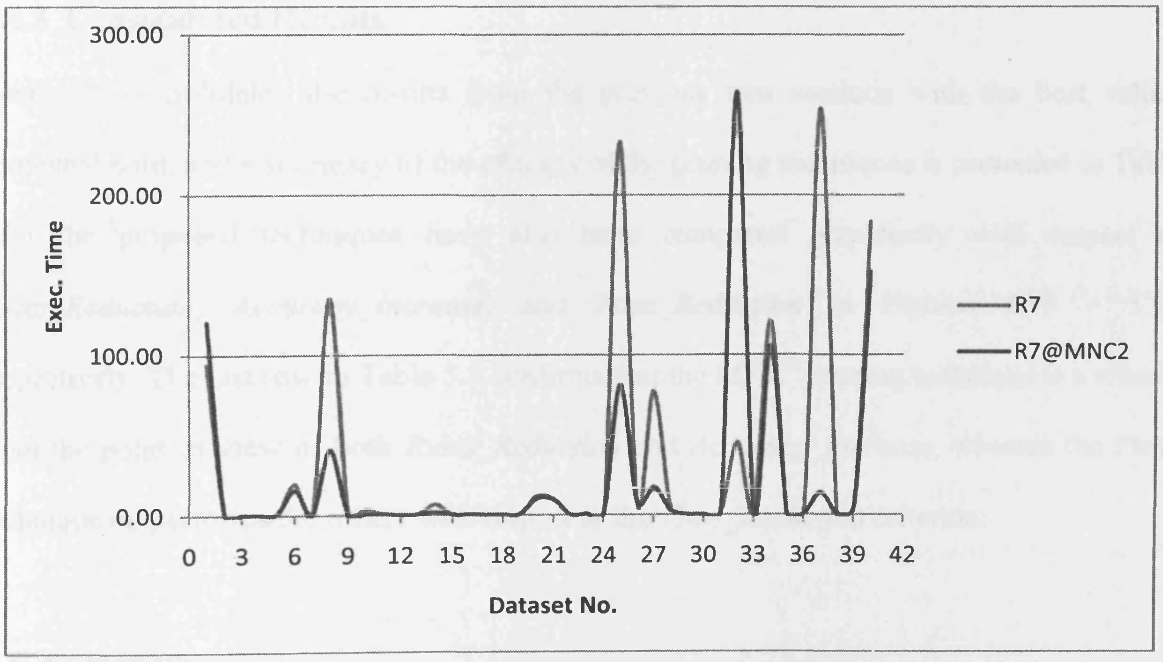


Figure 5.11 RULES-7@MNC^2 vs. RULES-7 in terms of Exec. Time

5.4.2 RULES-7@PMC vs. RULES-7

Table 5.6 compares the results obtained from RULES-7 by activating the PMC pruning technique against those of RULES-7 without pruning, while Table 5.7 summarises the results for RULES-7@PMC. A graphical representation of this table is also presented in Figures 5.12 – 5.14. The last row of Table 5.6 shows that the cumulative number of rules and the execution time reduces by 62.22% and 40.87% respectively, whereas the classification accuracy increases by 2.72%. Furthermore, it can be seen from Table 5.7 that for 63% of the datasets, there is an *overall improvement* in the model obtained by pruning. For 10% of the datasets, the PMC technique does not result in any accuracy improvement. The classification accuracy increases for 63% of the datasets and decreases for 27%. The number of rules obtained with PMC is lower for 93% and higher for none of the datasets tested, whereas the learning time reduces in case of 78% of the datasets and increases for only 12%.

5.4.3 Consolidated Results

Table 5.8 consolidates the results from the previous two sections with the best values formatted bold, and a summary of the efficacy of the pruning techniques is presented in Table 5.9. The proposed techniques have also been compared graphically with respect to *Rules_Reduction*, *Accuracy_Increase*, and *Time_Reduction* in Figures 5.15 – 5.17 respectively. The last row in Table 5.8 confirms that the MNC^2 pruning technique is a winner from the point of view of both *Rules_Reduction* and *Accuracy_Increase*, whereas the PMC technique outperforms the others with respect to the *Time_Reduction* criterion.

5.5 Summary

This chapter has initially proposed a new incremental post-pruning technique, MNC, for the RULES-7 algorithm in order to address the issue of overlapping native to the RULES family.

No.	Dataset	PMC € {1, 5, 10, 20, 30} %	No. of Rules		Redc. (%)	Accuracy (%)		Incr. (%)	Exec. Time		Redc. (%)	Speed Boost (*x*)	Impr. ?
			Rules7	Rules7 @PMC		Rules7	Rules7 @PMC		Rules7	Rules7 @PMC			
1	Arrythmia(M)L	1	71	66	**7.04**	74.00	73.00	**-1.35**	93.71	105.36	**-12.43**	**nil**	**no**
2	Balance-Scale(N)	5	113	60	**46.90**	58.83	63.33	**7.65**	0.14	0.10	**28.57**	**1.4x**	**yes**
3	Breast-Cancer(C)	20	25	19	**24.00**	93.48	92.90	**-0.62**	0.11	0.10	**9.09**	**1.1x**	**no**
4	Breast-Cancer(N)	30	74	18	**75.68**	66.07	69.64	**5.41**	0.22	0.07	**68.18**	**3.1x**	**yes**
5	Car(N)	10	94	9	**90.43**	61.35	74.85	**22.02**	0.75	0.09	**88.00**	**8.3x**	**yes**
6	Chess(N)L	5	48	17	**64.58**	94.21	95.41	**1.27**	19.30	7.10	**63.21**	**2.7x**	**yes**
7	Connectionist-Sonar(C)	10	28	27	**3.57**	83.00	84.00	**1.20**	0.65	0.63	**3.08**	**1.0x**	**yes**
8	*Cover-Type(M)L	30	446	35	**92.15**	58.34	68.14	**16.81**	134.96	8.98	**93.35**	**15.0x**	**yes**
9	Credit-Approval(M)	10	77	25	**67.53**	81.91	83.09	**1.44**	1.40	0.60	**57.14**	**2.3x**	**yes**
10	Cylinder-Bands(M)	30	88	26	**70.45**	61.89	67.55	**9.15**	3.40	1.08	**68.24**	**3.1x**	**yes**
11	Depression(M)	30	58	16	**72.41**	67.86	74.05	**9.12**	0.75	0.28	**62.67**	**2.7x**	**yes**
12	Ecoli(C)	1	23	23	**0.00**	85.81	85.81	**0.00**	0.05	0.06	**-20.00**	**nil**	**no**
13	Flags(M)	10	52	28	**46.15**	61.88	68.75	**11.11**	0.43	0.25	**41.86**	**1.7x**	**yes**
14	German-Credit(M)	10	240	62	**74.17**	71.00	72.00	**1.41**	6.45	2.10	**67.44**	**3.1x**	**yes**
15	Glass(C)	30	18	11	**38.89**	72.78	75.56	**3.82**	0.05	0.04	**20.00**	**1.3x**	**yes**
16	Heart-Cleveland(M)	20	69	18	**73.91**	56.07	61.43	**9.55**	0.45	0.15	**66.67**	**3.0x**	**yes**
17	Heart-Hungarian(M)	20	25	20	**20.00**	80.36	81.43	**1.33**	0.12	0.11	**8.33**	**1.1x**	**yes**
18	Hepatitis(M)	20	24	15	**37.50**	84.00	85.33	**1.59**	0.10	0.07	**30.00**	**1.4x**	**yes**
19	Horse-Colic(M)	20	49	18	**63.27**	76.39	76.67	**0.36**	0.51	0.25	**50.98**	**2.0x**	**yes**
20	Hyperthyroid(M)L	5	48	21	**56.25**	98.35	98.32	**-0.03**	11.51	5.39	**53.17**	**2.1x**	**no**
21	Hypothyroid(M)L	1	41	22	**46.34**	99.02	98.80	**-0.21**	9.70	4.40	**54.64**	**2.2x**	**no**
22	Image(C)	1	24	24	**0.00**	91.43	91.43	**0.00**	0.11	0.11	**0.00**	**nil**	**no**
23	Ionosphere(C)	10	33	30	**9.09**	92.06	93.82	**1.92**	0.41	0.39	**4.88**	**1.1x**	**yes**
24	Iris(C)	30	6	6	**0.00**	93.33	94.00	**0.71**	0.00	0.00	**0.00**	**nil**	**no**
25	Landsat(C)L	1	647	251	**61.21**	85.36	84.10	**-1.48**	233.09	130.54	**44.00**	**1.8x**	**no**
26	Mushroom(N)L	10	26	16	**38.46**	99.56	98.77	**-0.79**	8.01	5.88	**26.59**	**1.4x**	**no**
27	Nursery(N)L	20	357	16	**95.52**	65.73	76.81	**16.86**	76.80	4.40	**94.27**	**17.5x**	**yes**
28	Page-Blocks(C)L	1	88	74	**15.91**	92.50	92.15	**-0.38**	4.69	5.59	**-19.19**	**nil**	**no**
29	Parkinsons(C)	1	21	20	**4.76**	84.44	87.22	**3.29**	0.15	0.15	**0.00**	**nil**	**yes**
30	P-O-Patient(M)	30	28	11	**60.71**	67.50	73.75	**9.26**	0.03	0.02	**33.33**	**1.5x**	**yes**
31	Promoters(N)	10	20	17	**15.00**	84.00	84.00	**0.00**	0.17	0.16	**5.88**	**1.1x**	**yes**
32	Shuttle(C)L	10	99	25	**74.75**	99.83	99.69	**-0.13**	44.90	12.79	**71.51**	**3.5x**	**no**
33	Soybean-Large(N)	1	54	45	**16.67**	90.16	88.13	**-2.25**	1.79	1.61	**10.06**	**1.1x**	**no**
34	Spambase(C)L	10	157	42	**73.25**	90.54	89.93	**-0.67**	121.11	73.17	**39.58**	**1.7x**	**no**
35	Spect(C)	30	15	14	**6.67**	82.31	82.31	**0.00**	0.65	0.68	**-4.62**	**nil**	**no**
36	SPECT-Heart(N)	10	34	33	**2.94**	82.69	81.92	**-0.93**	0.45	0.46	**-2.22**	**nil**	**no**
37	Splice(N)L	1	254	146	**42.52**	90.35	91.03	**0.75**	253.92	191.14	**24.72**	**1.3x**	**yes**
38	Tic-Tac-Toe(N)	5	27	23	**14.81**	92.74	98.84	**6.58**	0.23	0.23	**0.00**	**nil**	**yes**
39	Vehicle(C)	5	130	59	**54.62**	68.78	69.15	**0.53**	3.46	1.86	**46.24**	**1.9x**	**yes**
40	Waveform-v2(C)L	1	718	273	**61.98**	80.89	81.11	**0.27**	183.60	153.95	**16.15**	**1.2x**	**yes**
Average:			111	42	**62.22**	80.52	82.71	**2.72**	30.46	18.01	**40.87**		

Table 5.6 RULES-7@PMC vs. RULES-7

No.	Criterion	No. of Datasets for which					
		Redc.		Equal		Incr.	
		Total	%age	Total	%age	Total	%age
1	No. of Rules	37	93%	3	7%	0	0%
2	Accuracy	11	27%	4	10%	25	63%
3	Exec. Time	31	78%	4	10%	5	12%
4	Overall Impr.	25					63%

Table 5.7 Summary of RULES-7@PMC

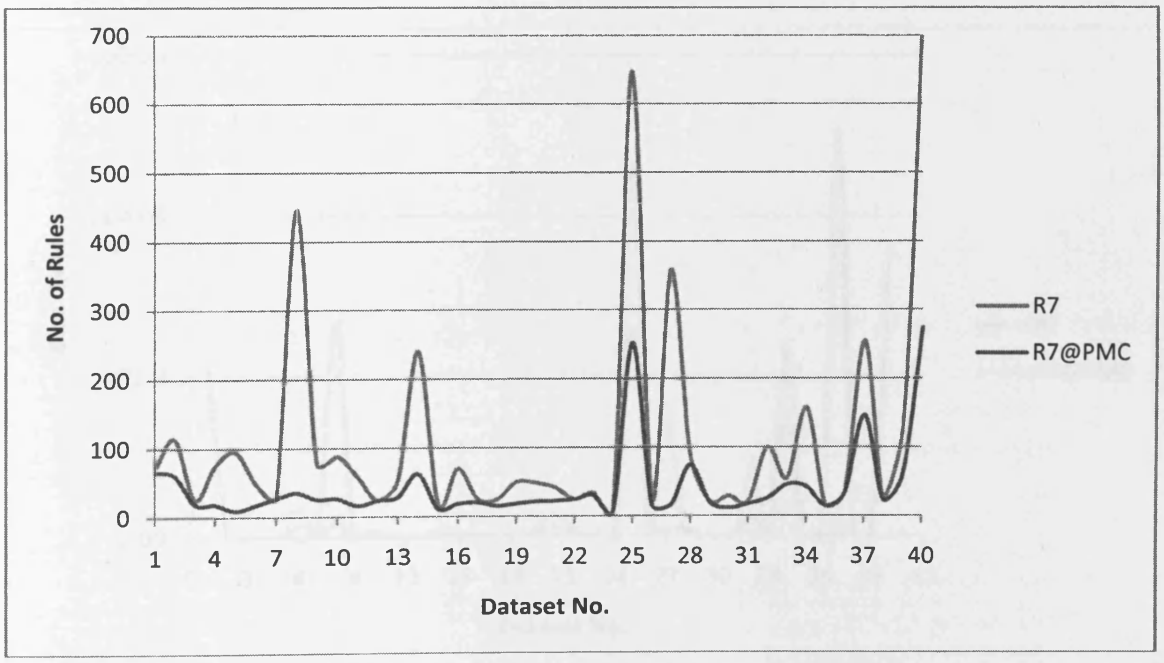


Figure 5.12 RULES-7@PMC vs. RULES-7 in terms of Number of Rules

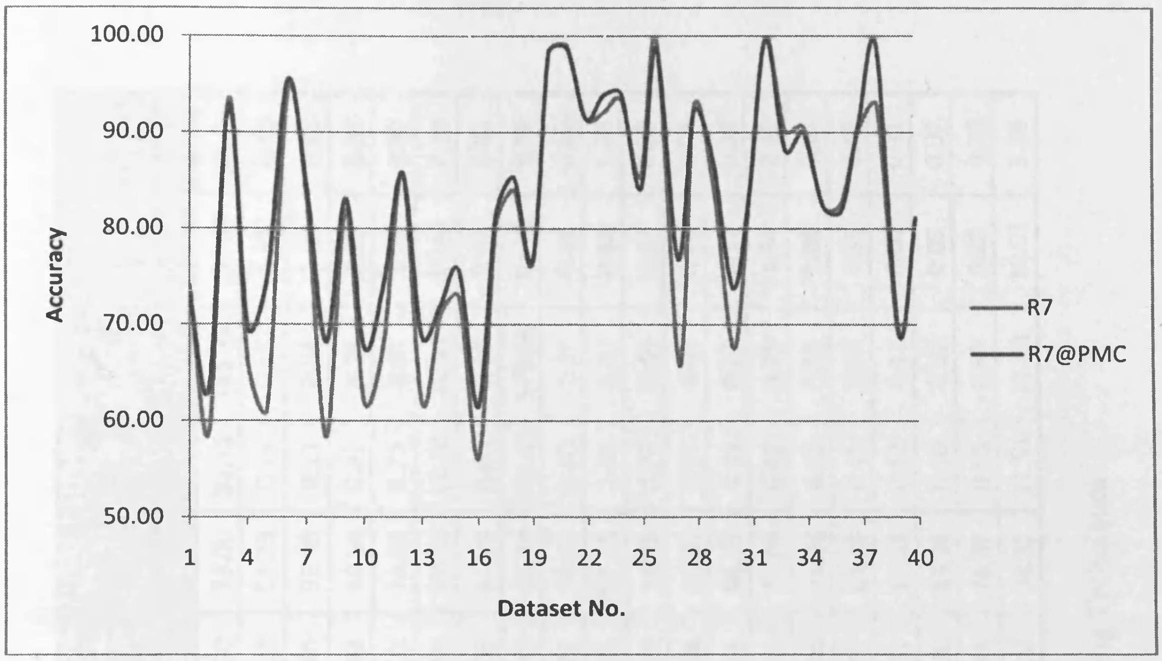


Figure 5.13 RULES-7@PMC vs. RULES-7 in terms of Accuracy

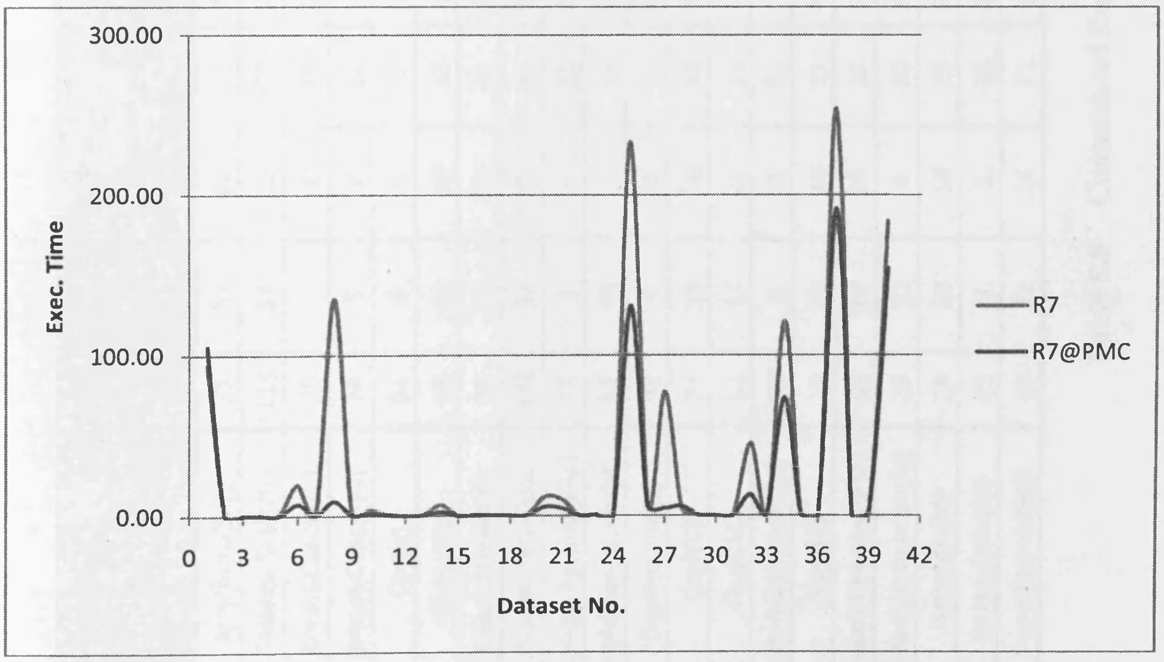


Figure 5.14 RULES-7@PMC vs. RULES-7 in terms of Exec. Time

No.	Dataset	No. of Rules				Accuracy (%)				Exec. Time (sec)			
		R7	R7 @MNC^{1}	R7 @MNC^{2}	R7 @PMC	R7	R7 @MNC^{1}	R7 @MNC^{2}	R7 @PMC	R7	R7 @MNC^{1}	R7 @MNC^{2}	R7 @PMC
1	Arrythmia(M)L	71	57	**32**	66	74.00	76.25	**81.50**	73.00	**93.71**	145.35	120.29	105.36
2	Balance-Scale(N)	113	**17**	19	60	58.83	**75.17**	71.33	63.33	0.14	0.17	**0.09**	0.10
3	Breast-Cancer(C)	25	3	**2**	19	93.48	95.22	**95.80**	92.90	0.11	**0.04**	0.15	0.10
4	Breast-Cancer(N)	74	5	**2**	18	66.07	71.79	**73.93**	69.64	0.22	0.24	**0.03**	0.07
5	Car(N)	94	**4**	**4**	9	61.35	74.97	**75.03**	74.85	0.75	0.83	1.03	**0.09**
6	Chess(N)L	48	21	26	**17**	94.21	**96.86**	96.67	95.41	19.30	26.17	15.44	**7.10**
7	Connectionist(C)	28	28	28	**27**	83.00	83.00	83.00	**84.00**	0.65	0.67	0.67	**0.63**
8	*Cover-Type(M)L	446	**32**	87	35	58.34	61.89	62.41	**68.14**	134.96	576.68	40.70	**8.98**
9	Credit-Approval(M)	77	5	**4**	25	81.91	**85.44**	82.21	83.09	1.40	1.57	**0.19**	0.60
10	Cylinder-Bands(M)	88	48	**3**	26	61.89	65.85	67.36	**67.55**	3.40	3.50	**0.42**	1.08
11	Depression(M)	58	**4**	25	16	67.86	70.95	73.33	**74.05**	0.75	0.40	0.37	**0.28**
12	Ecoli(C)	23	**19**	20	23	85.81	**86.45**	**86.45**	85.81	**0.05**	0.06	0.06	0.06
13	Flags(M)	52	**17**	25	28	61.88	67.50	66.88	**68.75**	0.43	0.26	0.31	**0.25**
14	German-Credit(M)	240	**8**	61	62	71.00	**72.10**	70.80	72.00	6.45	9.85	**1.54**	2.10
15	Glass(C)	18	20	20	**11**	72.78	74.44	74.44	**75.56**	0.05	0.08	0.08	**0.04**
16	Heart-Cleveland(M)	69	67	**14**	18	56.07	58.57	61.07	**61.43**	0.45	0.58	0.35	**0.15**
17	Heart-Hungarian(M)	25	22	**4**	20	80.36	82.14	**83.57**	81.43	0.12	0.12	**0.00**	0.11
18	Hepatitis(M)	24	18	**13**	15	84.00	84.00	84.00	**85.33**	0.10	0.10	**0.05**	0.07
19	Horse-Colic(M)	49	4	**2**	18	76.39	**83.06**	79.44	76.67	0.51	0.60	**0.20**	0.25
20	Hyperthyroid(M)L	48	**12**	26	21	98.35	**98.48**	98.10	98.32	11.51	21.13	10.01	**5.39**

Table 5.8 Consolidated Results for New Pruning Techniques

No.	Dataset	No. of Rules				Accuracy (%)				Exec. Time (sec)			
		R7	R7 @MNC^1	R7 @MNC^2	R7 @PMC	R7	R7 @MNC^1	R7 @MNC^2	R7 @PMC	R7	R7 @MNC^1	R7 @MNC^2	R7 @PMC
21	Hypothyroid(M)L	41	**8**	25	22	**99.02**	98.30	98.51	98.80	9.70	14.49	9.08	**4.40**
22	Image(C)	24	18	**17**	24	91.43	**93.33**	91.90	91.43	0.11	0.09	**0.07**	0.11
23	Ionosphere(C)	33	20	**12**	30	92.06	92.35	93.53	**93.82**	0.41	0.36	**0.33**	0.39
24	Iris(C)	6	**4**	**4**	6	93.33	**96.00**	**96.00**	94.00	**0.00**	**0.00**	**0.00**	**0.00**
25	Landsat(C)L	647	**131**	179	251	**85.36**	84.28	83.40	84.10	233.09	425.51	**80.84**	130.54
26	Mushroom(N)L	26	**15**	16	16	**99.56**	97.51	97.82	98.77	8.01	6.55	6.74	**5.88**
27	Nursery(N)L	357	81	71	**16**	65.73	68.76	75.84	**76.81**	76.80	**0.00**	16.94	4.40
28	Page-Blocks(C)L	88	26	**8**	74	92.50	90.86	**93.60**	92.15	4.69	5.99	**2.85**	5.59
29	Parkinsons(C)	21	17	**4**	20	84.44	83.89	85.56	**87.22**	0.15	0.15	**0.05**	0.15
30	P-O-Patient(M)	28	**3**	4	11	67.50	72.50	**75.00**	73.75	0.03	0.03	**0.01**	0.02
31	Promoters(N)	20	20	**9**	17	84.00	84.00	**87.00**	84.00	0.17	0.18	**0.06**	0.16
32	Shuttle(C)L	99	48	26	**25**	**99.83**	99.01	97.81	99.69	44.90	140.18	264.36	**12.79**
33	Soybean-Large(N)	54	**35**	36	45	90.16	90.94	**91.25**	88.13	1.79	1.56	**1.15**	1.61
34	Spambase(C)L	157	43	53	**42**	90.54	85.71	**92.28**	89.93	121.11	238.40	105.56	**73.17**
35	Spect(C)	15	**6**	12	14	82.31	83.85	**85.77**	82.31	0.65	0.54	**0.53**	0.68
36	SPECT-Heart(N)	34	**3**	4	33	82.69	**87.31**	86.92	81.92	0.45	0.42	**0.07**	0.46
37	Splice(N)L	254	77	**18**	146	90.35	90.00	**92.64**	91.03	253.92	226.64	**13.45**	191.14
38	Tic-Tac-Toe(N)	27	**21**	23	23	92.74	97.16	97.37	**98.84**	0.23	0.20	**0.18**	0.23
39	Vehicle(C)	130	81	**30**	59	68.78	**69.76**	69.15	69.15	3.46	4.34	2.80	**1.86**
40	Waveform-v2(C)L	718	**104**	139	273	80.89	79.97	78.37	**81.11**	183.60	775.78	**151.73**	153.95
Average:		111	29	**28**	42	80.52	82.74	**83.43**	82.71	30.46	65.75	21.22	**18.01**

Table 5.8 Consolidated Results for New Pruning Techniques (continued).

No.	Pruning Technique	No. of Datasets for which					
		Min No. of Rules		Max Accuracy		Min Exec. Time	
		Total	%age	Total	%age	Total	%age
1	MNC[1]	18	45%	12	30%	4	10%
2	MNC[2]	18	45%	14	35%	23	58%
3	PMC	6	15%	16	40%	17	43%

Table 5.9 Summary of New Pruning Techniques

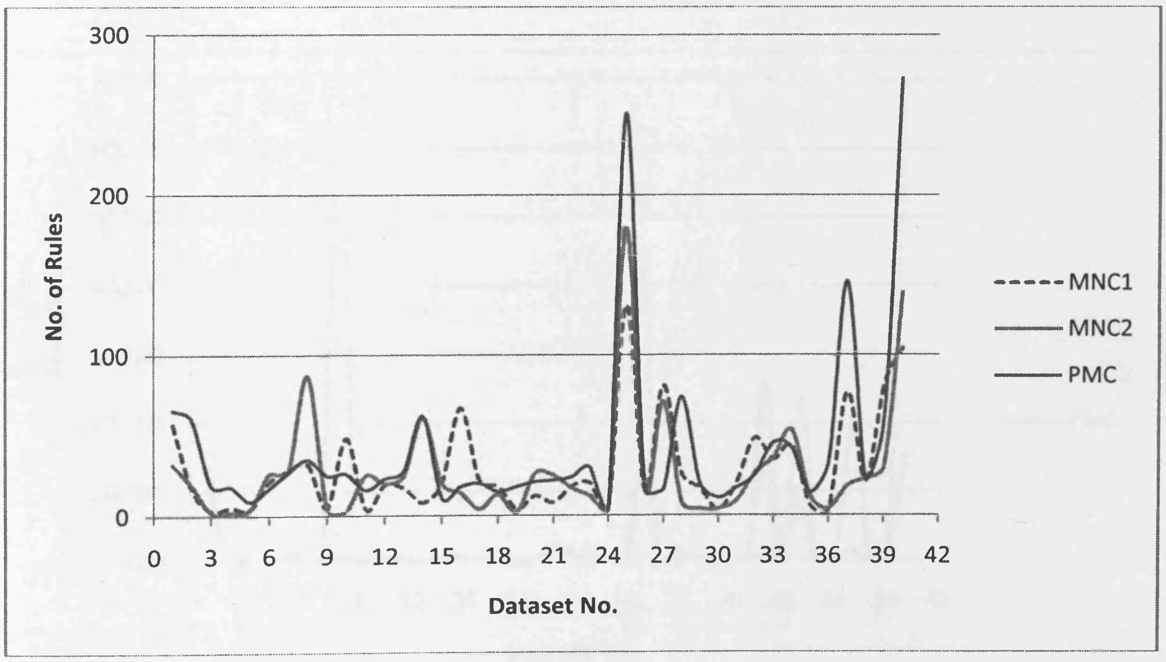


Figure 5.15 Comparison of Pruning Techniques in terms of Number of Rules

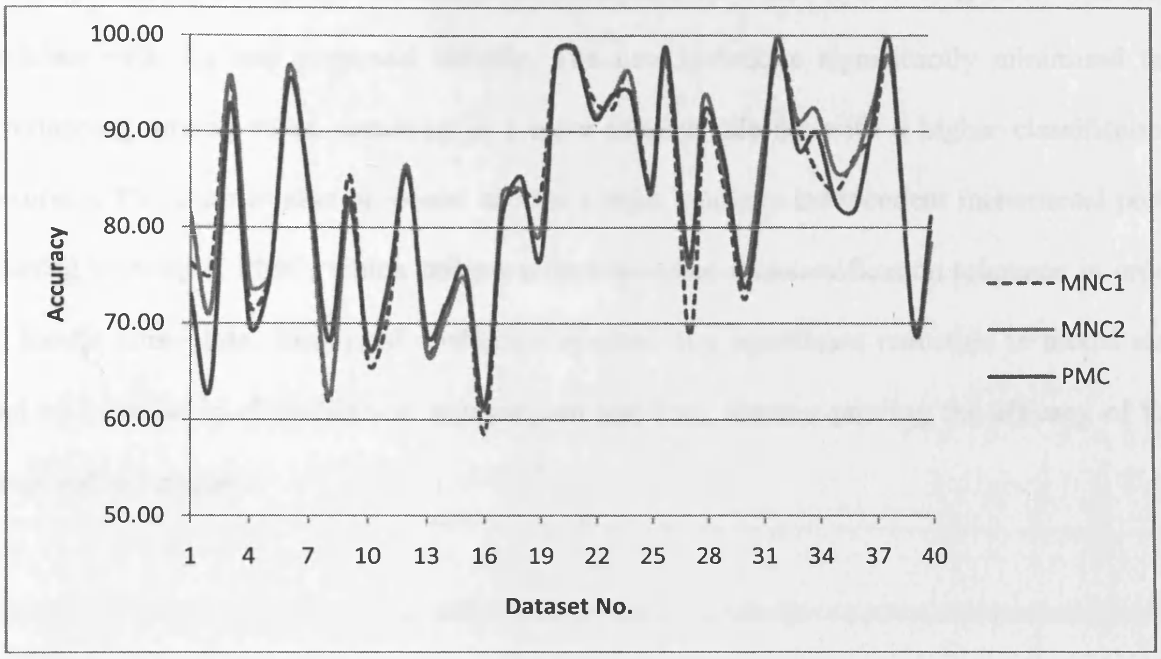


Figure 5.16 Comparison of Pruning Techniques in terms of Accuracy

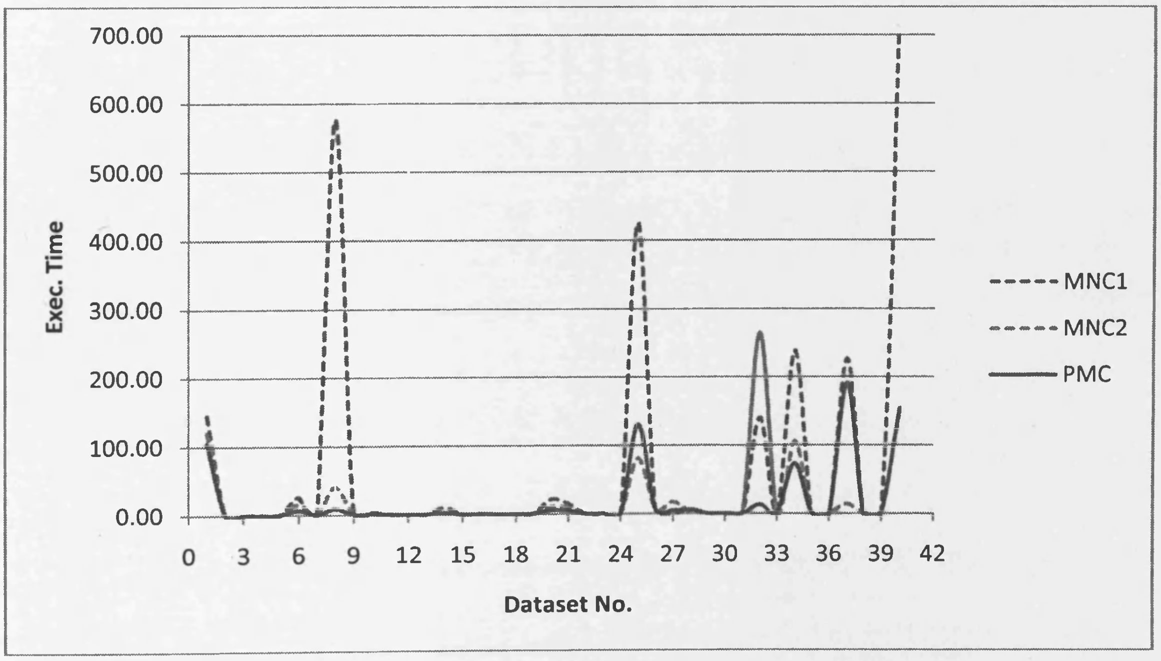


Figure 5.17 Comparison of Pruning Techniques in terms of Exec. Time

It then presented two hybrid versions of the same technique, the latter solving the inefficiency problem with the one proposed initially. The new technique significantly minimised the overlapping among rules, resulting in a more concise rule set with a higher classification accuracy. The chapter also proposed another simple heuristic-independent incremental post-pruning technique, PMC, which utilises a user-specified misclassification tolerance in order to handle noisy data. Empirical evaluation resulted in a significant reduction in model size and an increase in classification accuracy on test data, thereby proving the efficacy of the proposed techniques.

CHAPTER 6

CONCLUSION

This chapter summarises the work reported in this thesis. It highlights its key contributions, outlines the conclusions reached in this work and provides directions for future research.

6.1 Contributions

This research has focused on the extension of state-of-the-art methods developed to address the issues of scalability and robustness for classification rule induction algorithms. For this purpose, a critical review of existing scalable rule induction algorithms was carried out. This resulted in the development of new rule induction algorithms that can perform the task of 'data mining' in the true sense of the term, which is the ability to handle gigantic datasets as efficiently as possible. The proposed algorithms are also inherently capable of handling noisy data, which is an essential prerequisite for any learning algorithm that claims to be a sophisticated data mining tool. A detailed description of the contributions of this research is as follows:

- *A comprehensive analysis of scalable inductive learning techniques proposed to date.* An in-depth study of state-of-the-art inductive learning techniques was carried out, with special emphasis on scalability issues. The related field of association learning, as well as a relatively new area of associative classification, were also explored. This led to the discovery of rule mining constraints used in association learning along with an investigation of their suitability for classification learning.

- *A new classification rule induction algorithm for real-world data mining applications.* A sound critique of an existing algorithm RULES-6 was carried out, resulting in the discovery of several limitations. The identified issues were addressed and three new techniques were proposed to develop a new algorithm, RULES-7.

- *A new pre-processing discretisation technique for rule induction.* The new discretisation technique used a novel approach to discretisation, which discovers cut points for each class existing within an attribute. The class-centered approach resulted in the discovery of optimal cut points, because of which the classification accuracy increased significantly.

- *Two new simple pruning techniques for rule induction algorithms.* The main merit of the proposed new pruning techniques is the simplicity of their use. However, despite their simplicity, the techniques were able to cut down drastically on processing overheads. At the same time, the rule sets generated by these techniques were comprehensible as well as considerably accurate.

- *A thorough experimental evaluation to test the efficacy of the proposed techniques.* The techniques proposed in this research were evaluated thoroughly on a vast array of datasets, a great majority of which can truly be classified as 'large'. Furthermore, stratified 10-fold cross validation, which is widely accepted as the standard evaluation approach, was used throughout this research so as to ensure that the estimate of classification accuracy was as accurate as possible.

6.2 Conclusions

Due to some fundamental limitations of the decision tree learning paradigm, the development of rule induction algorithms started gaining importance within the machine learning community. These algorithms can address the issues with decision trees by mining a set of rules directly from the training data. However, despite their significant advantages, little work has been done to improve their scalability so as to make them suitable for data mining applications, in contrast with decision tree algorithms. This study has carried out a critical appraisal of existing scalable classification rule induction algorithms with the intent to address any weaknesses that may be discovered. It has also explored the closely related area of association learning in order to identify the potentially useful constraints for classification applications.

Chapter 3 studied in detail an existing rule induction algorithm RULES-6 which attempts to address the issues of scalability and noise-tolerance inherent with its predecessor RULES-3 Plus. The RULES-3 Plus algorithm was designed for small-scale industrial applications. It did not use any pruning techniques and instead concentrated on generating consistent classification rules. Because of this, the algorithm was unsuitable for data mining applications in that it could not handle large noisy datasets. The RULES-6 algorithm attempted to fix these issues by introducing a search space pruning control structure which could declare some attribute values invalid based on certain user specified criteria. However, several limitations of the control structure were identified which adversely affected the algorithm's scalability and noise handling capabilities. Along with this, two other issues were also identified and addressed, namely duplicate candidate rule generation as well as the specialisation measure parameter assumption. Finally, three new techniques were proposed for RULES-7, two of which were based on ideas from association rule mining. The empirical evaluation of RULES-7 against RULES-6 proved that the new algorithm was not only many orders of magnitude faster than its predecessor but also produced rule sets that were significantly accurate.

Chapter 4 proposed a new pre-processing discretisation technique for the RULES-7 algorithm. The proposed technique was based on the MDLP-entropy discretisation technique which is regarded as the most accurate within the data mining community in terms of classification accuracy. The proposed discretisation technique EDISC however used a novel concept, referred to as scope of classes. It used the starting and ending points of a class within an attribute to determine its scope in that particular attribute. The scope limited list was then used for discovering the optimal cut points for each class existing within the attribute. The technique was tested experimentally and was found to be considerably more accurate than the usual entropy discretisation technique.

Chapter 5 proposed two new simple pruning techniques for use with RULES-7. The first of these was aimed at addressing the issue of overlapping inherent to the RULES family of algorithms in order to reduce the total number of rules. Experimental evaluation confirmed that the new technique drastically reduced the amount of overlapping, resulting in a much more concise rule set with better classification accuracy. The second technique was specifically designed to handle the issue of noisy data and was based on the use of a misclassification tolerance. This technique also gave significant improvements in classification accuracy for a large number of the tested datasets along with reducing the total number of rules.

6.3 Future Research Directions

The pruning techniques proposed for RULES-7 in the first part of this work have their foundations on the pre-pruning paradigm. Possible areas for further work can include the development of additional pre-pruning as well as some effective post-pruning techniques designed to simplify the final rule set. Furthermore, hybrid pruning techniques might also be considered in order to gain the benefits of both pre-pruning speed and post-pruning accuracy. In any case, the right strategy lies in finding the right balance between generality and complexity so as to increase the classification accuracy. In order to accelerate support counting, efficient data structures as well as ideas from closed and maximal frequent itemset mining could also be used.

Using the idea of the scope of classes introduced in the discretisation technique proposed in the second part of this work, new offline as well as online discretisation techniques may be attempted. Since the new discretisation technique is an overlapping one, the idea might be extended to fuzzy discretisation, where a value may belong to multiple intervals identified for a particular class, each with a certain degree of membership. An alternative discretisation approach such as splitting or merging might also be adopted in order to exploit the class-centered methodology described in this work.

It may also be possible to extend the RULES-7 algorithm proposed in this work in several possible ways. This may involve the use of a higher-level representation language, development of new feature selection techniques, as well as prior knowledge about the domain to restrict the search space. Methods of constructive induction may also be used in order to construct new features from those already available so as to improve the generalisation capability of the learning algorithm. The use of sampling or partitioning methods may be attempted in case the data does not fit in the main memory. Furthermore, bagging and boosting techniques may be used with the algorithm in order to further improve its classification accuracy. Finally, RULES-7 may be adapted to perform regression instead of classification.

APPENDIX A

DESCRIPTION OF DATASETS

With the exception of the 'Depression' dataset obtained from Williams College (Veaux, 2007), all the datasets used in this work have been downloaded from the University of California at Irvine (UCI), repository of machine learning databases (Blake and Merz, 1998). What follows is a description of these datasets in terms of the domain from which they have been taken and the classification objective in case of each.

Adult(M)L: This dataset was extracted from the 1994 Census database. It consists of attributes such as age, work class, education, marital status, occupation etc and the task is to predict whether a person in the US makes over $50,000.

Anneal(M): This dataset involves classification of steel into different annealing treatment groups based on attributes such as its family, type, carbon content, hardness, formability etc. The dataset comprises a total of six classes and has lots of missing values.

Arrhythmia(M)L: This dataset is from the medical domain and is a collection of patient records. The task is to identify the presence or absence of cardiac arrhythmia in a patient and to classify it in one of 16 classes. Class 1 refers to 'normal' ECG and indicates the absence of arrhythmia whereas classes 2 to 15 refer to different classes of arrhythmia. Class 16 refers to the rest of the records that remain unclassified.

Balance-Scale(N): This dataset was generated to model psychological experimental results.

Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The attributes are the left weight, the left distance, the right weight, and the right distance.

Blood-Transfusion(C): This dataset is from the "Blood Transfusion Service Center" in Taiwan. It uses of a blood donor such as Recency (months since last donation), Frequency (total number of donation) etc to predict whether he/she donated blood in March 2007.

Breast-Cancer(C): This is a medical dataset that includes attributes from a digitised image of a breast mass's fine needle aspirate (FNA) such as radius, texture, perimeter, area etc. The task is to predict whether breast cancer is malignant or benign.

Breast-Cancer(N): This is similar to the continuous type Breast-Cancer data except that it uses different attributes such as age, menopause, tumor size etc and the task is to classify into one of two categories, i.e. no-recurrence-events or recurrence-events.

Car(N): This dataset evaluates cars according to attributes such as price, technical characteristics, safety etc. The classification task involves grouping into one of 4 classes (unacc, acc, good and vgood).

Chess(N)L: This dataset includes examples each of which corresponds to a board description for the chess endgame. The task is to predict whether the configuration falls in the "win" or "no-win" class.

Connect-4(N)L: This dataset contains all legal 8-ply positions in the game of connect-4 in which neither player has won yet, and in which the next move is not forced. The task is to classify into one of three classes (win, loss, and draw).

Connectionist-Bench(C): This dataset involves the classification of sonar signals using a neural network. Each example is described by a set of 60 features in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock.

Cover-Type(M)L: This dataset involves predicting the type of the forest cover out of the 7 possible classes using attributes such as elevation, aspect, slope etc.

Credit-Approval(M): This dataset is from the banking and finance industry and involves accepting or rejecting credit card applications based on a total of 15 attributes.

Cylinder-Bands(M): This dataset is from the manufacturing industry and involves the identification of the band type in rotogravure printing given attributes such as timestamp, cylinder number, customer, job number etc.

Depression(M): This is a medical dataset which predicts the presence or absence of depression in a patient given features such as sex, age, ethorf, cepdura etc.

Dermatology(M): This is a medical dataset that involves the differential diagnosis of erythemato-squamous diseases in dermatology. It includes features such as erythema, scaling, definite borders, itching etc. The task is to classify the disease into one of 6 groups (psoriasis, seboreic dermatitis, lichen planus, pityriasis rosea, cronic dermatitis, and pityriasis rubra pilaris).

Echocardiogram(M): This is a medical dataset that predicts whether or not the patient will survive at least one year given attributes such as still-alive, age-at-heart-attack, pericardial-effusion etc.

Ecoli(C): This dataset is from the medical domain and involves classification of the type of Ecoli into one of eight classes given features such as sequence name, mcg, gvh, lip etc.

Flags(M): This dataset includes features of different countries such as name, landmass, zone, area, population etc along with the features in their flags such as the number of stripes, the number of colors, and the presence or absence of different colors etc. Multiple attributes may be chosen as classification attributes such as the religion of a country, the colors in its flag etc. The classification feature used in this study is the religion of the country.

German-Credit(M): This dataset identifies whether people are credit risks or not based on their loan applications. The loan applications include features such as status-account, duration, credit-history, purpose, credit-amount etc.

Glass(C): This dataset is a collection of records from crime lab reports in which the task is to classify the glass into one of seven types based on features such as refractive index, sodium content, magnesium content etc..

Hayes-Roth(N): This dataset is named after its creators and classifies examples into one of 3 classes based on features such as name, hobby, age, educational level etc.

Heart-Cleveland(M): This is a medical dataset with attributes such as age, sex, cp, trestbps, cholesterol etc and the classification task involves the diagnosis of the heart disease into one of 5 classes.

Heart-Hungarian(M): This is similar to the Heart-Cleveland dataset except that the goal is to identify only the presence or absence of heart disease.

Hepatitis(M): This is a medical dataset with attributes such as age, sex, steroid, antivirals, fatigue etc relating to the hepatitis disease and the task is to predict whether a person will die or live.

Horse-Colic(M): This dataset includes attributes related to horses such as their age, pulse, rectal temperature etc in order to classify whether a lesion is surgical or not.

Hyperthyroid(M)L: This dataset predicts the presence or absence of the hyperthyroid disease and sorts it into one of 3 classes (goitre, hyperthyroid, and T3-toxic). The attributes include age, sex, on-thyroxine, query-on-thyroxine, on-antithyroid-medication, sick, pregnant etc.

Hypothyroid(M)L: This dataset is similar to the Hyperthyroid data except that it predicts the presence or absence of the hypothyroid disease and groups it into one of 3 classes (comp-hypothyroid, pri-hypothyroid, and sec-hypothyroid).

Image(C): This dataset includes examples drawn randomly from a database of 7 outdoor images. The images were handsegmented to create a classification for every pixel. The task is to classify the images into one of 7 classes namely brickface, sky, foliage, cement, window, path, and grass given features such as region-centroid-col, region-centroid-row, region-pixel-count, short-line-density etc.

Ionosphere(C): This dataset involves classification of radar returns from the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere.

Iris(C): This is probably the most well-known dataset in the machine learning literature. It includes examples described by 4 continuous attributes (sepal length, sepal width, petal length, and petal width). The task is to identify the type of the Iris from one of 3 classes (Iris Setosa, Iris Versicolour, and Iris Virginica).

Landsat(C)L: This dataset consists of the multi-spectral values of pixels in 3x3 neighbourhoods in a satellite image along with the classification associated with the central pixel in each neighbourhood. The task is to predict this classification, given features such as red soil, cotton crop, grey soil, damp grey soil etc.

Letter(C)L: This dataset involves identification of each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. The dataset includes attributes such as horizontal position of box, vertical position of box, width of box, height of box etc.

Lymphography(N): This dataset involves diagnosis of the different diseases of lymph nodes as per their conditions in those diseases corresponding to 4 classes (normal find, metastases, malign lymph, and fibrosis). The attributes include lymphatics, block of affere, bl. of lymph., bl. of lymph etc.

Magic(C)L: This dataset is MC generated to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique. The task is to classify into one of 2 classes namely gamma and hadron given features such as fLength, fWidth, fSize, fConc etc.

Mushroom(N)L: This dataset comprises mushroom samples with features such as cap-shape, cap-surface, cap-color, the presence or absence of bruises, odor etc. The task is to classify a mushroom as either edible or poisonous.

Nursery(N)L: This dataset was used during several years in 1980's when there was excessive enrollment to nursery schools in Ljubljana, Slovenia, etc and the rejected applications frequently needed an objective explanation. The task is to classify an application into one of 4 categories (not_recom, priority, recommend, spec_prior, and very_recom). The given attributes include parents, has_nurs, form, children, housing etc.

Optical(C)L: This dataset involves optical recognition of handwritten digits into one of 10 classes given a total of 64 features with all of them having integer values.

Ozone(C)L: This dataset involves ozone level detection with the objective of classifying into one of 2 classes namely ozone day and normal day. All 72 attributes in the dataset are continuous valued.

Page-Blocks(C)L: This dataset involves classification of page blocks such that each example in the dataset represents one block and comes from 54 distinct documents. The attributes include height, length, area, eccentricity etc.

Parkinsons(C): This dataset has been constructed from a range of biomedical voice measurements from 31 people of which 23 had the Parkinson's disease (PD). Each example in the dataset represents a particular voice measure, and each attribute corresponds to one of 195 voice recording from these individuals. The task is to discriminate healthy people from those with PD, according to the "status" attribute which is 0 for healthy and 1 for PD.

Pendigits(C)L: This dataset was created by collecting 250 samples of pen-based handwritten digits from 44 writers. The task is to classify these handwritten digits into one of 10 classes using 16 attributes.

Pima-Indians(C): This is a medical dataset that involves predicting whether diabetes is present in a patient or not. All the patients were females of at least 21 years age. The features used for prediction include number of times pregnant, plasma glucose concentration, diastolic blood pressure etc.

Post-Operative-Patient(M): The classification task in this dataset is to determine whether patients in a postoperative recovery area should be sent to next. Because hypothermia is a significant concern after surgery, the attributes correspond roughly to body temperature measurements. The classification attribute has three values (patient sent to ICU, patient prepared to go home, and patient sent to general hospital floor).

Promoters(N): This dataset represents nucleotides of the DNA sequence (a, t, c or g). The task is to predict whether a sequence is in a promoter region (+) or not (-).

Shuttle(C)L: This dataset is described by attributes such as rad flow, fpv close, fpv open, high etc and the task is to classify shuttle landing into one of 7 classes represented by integers 1 - 7.

Soybean-Large(N): This dataset involves diagnosis of soybean disease into one of 19 classes based on attributes such as date, plant-stand, precip, temp, hail, crop-hist etc.

Spambase(C)L: This dataset includes a total of 57 continuous attributes and the task is to predict whether an e-mail message should be classified as spam or not-spam.

Spect(C): This dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories (normal and abnormal). The database of 267 SPECT image sets (patients) was processed to extract features that summarise the original SPECT images.

SPECT-Heart(N): This dataset is similar to the Spect data with the difference that it has only 22 attributes all of which are binary.

Splice(N)L: This dataset includes 60 attributes. Each example is described by its name and the sequential DNA nucleotide positions. The task is to identify the 3 classes (donors, acceptors, and neither).

Tic-Tac-Toe(N): This database encodes the complete set of possible board configurations at the end of tic-tac-toe games, where *x* is assumed to have played first. The task is to predict whether *x* wins or not.

Vehicle(C): The task in this dataset is to classify a given silhouette as one of four types of vehicle (bus, opel, saab, and van). This is accomplished using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles.

Waveform-v2(C)L: The goal in this dataset is to classify the waves into one of 3 classes given 40 different attributes all of which are continuous valued.

Wine(C): This dataset is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The task is to predict the type of the wine (out of a total number of 3 classes) given attributes corresponding to the 13 constituents.

Yeast(C): This dataset comprises a total of 8 continuous attributes. The task is to predict the localisation site out of a set of 10 class values.

APPENDIX B

THE RULE-3 PLUS ALGORITHM

```
Step 1.  Quantize attributes that have numerical values.
Step 2.  Select an unclassified example and form array SETAV.
Step 3.  Initialise arrays PRSET and T_PRSET (PRSET and T_PRSET will consist
         of $m_{PRSET}$ expressions with null conditions and zero H measures) and set $n_{co} = 0$.
Step 4.  IF $n_{co} < n_a$
         THEN $n_{co} = n_{co} + 1$ and set $m = 0$;
         ELSE the example itself is taken as a rule and go to Step 7.
Step 5.  DO
             $m = m + 1$;
             Form an array of expressions (T_EXP). The elements of this array are
             combinations of expression m in PRSET with conditions from SETAV
             that differ from the conditions already included in the expression m
             (the number of elements in T_EXP is: $n_a - n_{co}$. Set $k = 1$;)
             DO
                 $k = k + 1$;
                 Compute the H measure of expression k in T_EXP;
                 IF its H measure is higher than the H measure of any
                 expression in T_PRSET
                 THEN replace the expression having the lowest H measure
                 with expression k;
             WHILE $k < n_a - n_{co}$;
             Discard the array T_EXP;
  WHILE $m < m_{PRSET}$.
```

A pseudo-code description of RULES-3 Plus (Pham and Dimov, 1997b)

```
Step 6.  IF there are consistent expressions in T_PRSET
         THEN choose as a rule the expression that has the highest H measure and
         discard the others;
              mark the examples covered by this rule as classified;
              go to Step 7;
         ELSE copy T_PRSET into PRSET;
              initialise T_PRSET and go to Step 4.
Step 7.  IF there are no more unclassified examples
         THEN STOP;
  ELSE go to Step 2.
```

A pseudo-code description of RULES-3 Plus (continued).

APPENDIX C

THE SRI ALGORITHM

INDUCE_RULES PROCEDURE

```
Procedure Induce_Rules (TrainingSet, BeamWidth)
RuleSet = $\emptyset$                                                          (step 1)
For each class in the TrainingSet Do                                           (step 2)
  Instances = TrainingSet                                                      (step 3)
  While Positive (Instances) $\neq \emptyset$ Do                               (step 4)
    Rule = Induce_One_Rule (Instances, CurrentClass, BeamWidth)                (step 5)
    If Rule_Generation_Stopping_Criterion (Rule, Instances) is True Then       (step 6)
      Exit While
    Instances = Instances – Covered_Positives (Rule, Instances)                (step 7)
    RuleSet = RuleSet $\cup$ {Rule}                                            (step 8)
  End While
End For
Return RuleSet                                                                 (step 9)
End                                                                            (step 10)
```

A pseudo-code description of SRI (Pham and Afify, 2006a)

INDUCE_ONE_RULE PROCEDURE

```
Procedure Induce_One_Rule (Instances: ClassLabel: $w$)
PartialRules = NewPartialRules = $\emptyset$
BestRule = most general rule (the rule with no conditions)                (step 1)
PartialRules = PartialRules $\cup$ {BestRule}
While PartialRules $\neq \emptyset$ Do                                     (step 2)
  For each Rule $\in$ PartialRules Do
    For each nominal attribute $A_i$ that does not appear in Rule Do
      For each valid value $v_{ij}$ of $A_i \in$ Rule.ValidValues Do
        NewRule = Rule $\wedge [A_i = v_{ij}]$                              (step 3)
        NewRule.Instances = Covered_Instances (Rule.Instances, $v_{ij}$)    (step 4)
        If NewRule.Score > BestRule.Score Then                             (step 5)
          BestRule = NewRule
        If Covered_Positives (NewRule) < MinPositives OR                   (step 6)
          Covered_Negatives (Rule) – Covered_Negatives (NewRule) < MinNegatives OR
                                                                           (step 7)
          Consistency (NewRule) = 100% Then                                (step 8)
          Parent (NewRule).InvalidValues = Parent (NewRule).InvalidValues + {$v_{is}$}
                                                                           (step 9)
        Else
          NewPartialRules = NewPartialRules $\cup$ {NewRule}               (step 10)
      End For
    End For
  End For
Empty PartialRules
```

A pseudo-code description of the *Induce_One_Rule* procedure of SRI

```
  For each $Rule \in NewPartialRules$ Do
    If $Rule$.OptimisticScore $\leq$ $BestRule$.Score Then                    (step 11)
      $NewPartialRules = NewPartialRules - \{Rule\}$                            (step 12)
      Parent ($Rule$).InvalidValues = Parent ($Rule$).InvalidValues + Last_Value_Added ($Rule$)
                                                                               (step 13)
  End For
  For each $Rule \in NewPartialRules$ Do
    $Rule$.ValidValues = $Rule$.ValidValues − Parent ($Rule$).InvalidValues     (step 14)
  End For
  If $w > 1$ Then
    Remove from $NewPartialRules$ all duplicate rules
    Select $w$ best rules from $NewPartialRules$ and insert into $PartialRules$  (step 15)
    Remove all rules from $NewPartialRules$
End While
Return $BestRule$
End
```

A pseudo-code description of the *Induce_One_Rule* procedure of SRI (continued).

REFERENCES

Afify, A. A. (2004). *Design and Analysis of Scalable Rule Induction Systems.* Ph.D. Thesis, Systems Engineering Division, University of Wales, Cardiff, UK.

Agrawal, R., Imielinski, T. and Swami, A. (1993). Mining Association Rules between Sets of Items in Large Databases. *Proc. of ACM SIGMOD Conf.*, Washington D.C., pp. 207-216.

Agrawal, R. and Srikant, R. (1994). Fast Algorithms for Mining Association Rules. *Proc. of the 20th VLDB Conf.*, Santiago, Chile, pp. 487-499.

Aha, D. W., Kibler, D. and Albert, M. K. (1991). Instance-Based Learning Algorithms. *Machine Learning*, Vol. 6, pp. 37-66.

An, A. and Cercone, N. (1999). Discretization of Continuous Attributes for Learning Classification Rules. *Proc. of the 3rd Pacific-Asia Conf. on Methodologies for Knowledge Discovery and Data Mining*, pp. 509-514.

Bay, S. D. (2000). Multivariate discretization of continuous variables for set mining. *Proc. of the 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Boston, Massachusetts, United States, pp. 315-319.

Bayardo, R. J. (1998). Efficiently mining long patterns from databases. *Proc. of the 1998 ACM SIGMOD Int. Conf. on Management of Data*, Seattle, Washington, United States, pp. 85-93.

Bergadano, F., Matwin, S., Michalski, R. S. and Zhang, J. (1992). Learning Two-Tiered Descriptions of Flexible Concepts: The POSEIDON System. *Machine Learning*, Vol. 8, pp. 5-43.

Bigot, S. (2002). *New Techniques for Handling Continuous Values in Inductive Learning.* Ph.D. Thesis, Systems Engineering Division, University of Wales, Cardiff, UK.

Blake, C. L. and Merz, C. J. (1998) UCI Repository of machine learning databases. IN (Ed.^(Eds.) *Department of Information and Computer Science.* ed. Irvine, CA, University of California.

Breiman, L., Friedman, J., Stone, C. J. and Olshen, R. A. (1984). *Classification and Regression Trees*, Wadsworth, Belmont, CA.

Breslow, L. and Aha, D. (1997). Simplifying decision trees: A survey. *Knowl. Eng. Rev.*, Vol. 12, pp. 1-40.

Brin, S., Motwani, R., Ullman, J. D. and Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, Tucson, Arizona, USA, pp. 255-264.

Brunk, C. A. and Pazzani, M. J. (1991). An Investigation of Noise-Tolerant Relational Concept Learning Algorithms. *Proc. of the 8th Int. Workshop on Machine Learning*, pp. 389-393.

Burdick, D., Calimlim, M. and Gehrke, J. (2001). MAFIA: a maximal frequent itemset algorithm for transactional databases. *Proc. of 17th Int. Conf. on Data Engineering*, pp. 443-452.

Catlett, J. (1991a). *Megainduction: Machine Learning on Very Large Databases*. Ph.D. Thesis, School of Computer Science, University of Technology, Sydney, Australia.

Catlett, J. (1991b). On Changing Continuous Attributes into Ordered Discrete Attributes. *Proc. of the European Working Session on Machine Learning*, pp. 164-178.

Cendrowska, J. (1987). PRISM: An algorithm for inducing modular rules. *Int. Journal of Man-Machine Studies*, Vol. 27, pp. 349-370.

Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning. *Proc. of the European Conf. on Artificial Intelligence*, pp. 147-149.

Clark, P. and Boswell, R. (1991). Rule induction with CN2: Some recent improvements. *Proc. of the 5th European Working Session on Learning*, Porto, Portugal, pp. 151-163.

Clark, P. and Niblett, T. (1989). The CN2 Induction Algorithm. *Machine Learning*, Vol. 3, pp. 261-283.

Cohen, W. W. (1993). Efficient pruning methods for separate-and-conquer rule learning systems. *Proc. of the 13th Int. Joint Conf. on Artificial Intelligence*, pp. 988-994.

Cohen, W. W. (1995). Fast Effective Rule Induction. *Proc. of the 12th Int. Conf. on Machine Learning*, pp. 115-123.

Dhar, V., Chou, D. and Provost, F. (2000). Discovering Interesting Patterns for Investment Decision Making with GLOWER - A Genetic Learner Overlaid with Entropy Reduction. *Data Min. Knowl. Discov.*, Vol. 4, pp. 251-280.

Dougherty, J., Kohavi, R. and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *Proc. of the 12th Int. Conf. on Machine Learning*, Tahoe City, California, USA, pp. 194-202.

Fayyad, U. M. and Irani, K. B. (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. *Proc. of the 13th Int. Joint Conf. on Artificial Intelligence*, pp. 1022-1027.

Fayyad, U. M., Piatetsky-Shapiro, G. and Smyth, P. (1996). From data mining to knowledge discovery: an overview. In: *Advances in knowledge discovery and data mining* (Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. and Uthurusamy, R., (Eds.)), Menlo Park, CA, American Association for Artificial Intelligence.

Furnkranz, J. (1994a). FOSSIL: a robust relational learner. *Proc. of the European Conf. on Machine Learning*, Catania, Italy, pp. 122-137.

Furnkranz, J. (1994b). Top-Down Pruning in Relational Learning. *Proc. of the 11th European Conf. on Artificial intelligence*, Amsterdam, The Netherlands, pp. 453-457.

Furnkranz, J. (1997). Pruning Algorithms for Rule Learning. *Machine Learning*, Vol. 27, pp. 139-172.

Furnkranz, J. (1999). Separate-and-Conquer Rule Learning. *Artif. Intell. Rev.*, Vol. 13, pp. 3-54.

Furnkranz, J. and Widmer, G. (1994). Incremental Reduced Error Pruning. *Proc. of the 11th Int. Conf. on Machine Learning*, New Brunswick, NJ, pp. 70-77.

Gama, J., Torgo, L. and Soares, C. (1998). Dynamic Discretization of Continuous Attributes. *Proc. of the 6th Ibero-American Conf. on AI: Progress in Artificial Intelligence*, Lisbon, Portugal, pp. 160-169.

Gehrke, J. (2006). Classification and Regression Trees. In: *Encyclopedia of Data Warehousing and Mining* (Brennan, E., Bubnis, A., Davies, R. and VanderHook, S., (Eds.)), Hershey, PA, Idea Group Publishing.

Gehrke, J., Ganti, V., Ramakrishnan, R. and Loh, W.-Y. (1999). BOAT - Optimistic Decision Tree Construction. *Proc. of the 1999 ACM SIGMOD Int. Conf. on Management of data*, Philadelphia, Pennsylvania, United States, pp. 169-180.

Gehrke, J., Ramakrishnan, R. and Ganti, V. (1998). RainForest - A Framework for Fast Decision Tree Construction of Large Datasets. *Proc. of the 24th Int. Conf. on Very Large Data Bases (VLDB)*, New York, USA, pp. 416-427.

Grzymala-Busse, J. W. and Shah, P. (2000). A Comparison of Rule Matching Methods Used in AQ15 and LERS. *Proc. of the 12th Int. Symposium on Foundations of Intelligent Systems*, pp. 148-156.

Han, J. and Kamber, M. (2006). *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, San Francisco, USA.

Han, J., Pei, J., Yin, Y. and Mao, R. (2000). Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Mining and Knowledge Discovery*, Vol. 8, pp. 53-87.

Han, J., Wang, J., Lu, Y. and Tzvetkov, P. (2002). Mining Top-K frequent closed patterns without minimum support. *Proc. of IEEE Conf. on Data Mining, ICDM*, pp. 211-218.

Ho, K. M. and Scott, P. D. (1997). Zeta: a global method for discretization of continuous variables. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9, pp. 718-730.

Holsheimer, M. and Siebes, A. (1991). Data Mining: The Search for Knowledge in Databases. *Technical Report*, CWI Centre for Mathematics and Computer Science, CS-R9406, Amsterdam, The Netherlands.

Holte, R. C. (1993). Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning*, Vol. 11, pp. 63-90.

Hong, J., Mozetic, I. and Michalski, R. S. (1986). AQ15 - Incremental Learning of Attribute-based Descriptions from Examples, the Method and User's Guide. *Reports of the Intelligent Systems Group, ISG 86-5*, Department of Computer Science, University of Illinois, Urbana.

Jain, A. K., Murty, M. N. and Flynn, P. J. (1999). Data Clustering: A Review. *ACM Computing Surveys*, Vol. 31, pp. 264-323.

Jianyong, W., Han, J., Lu, Y. and Tzvetkov, P. (2005). TFP: an efficient algorithm for mining top-k frequent closed itemsets. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, pp. 652-663.

Kalbfleish, J. (1979). *Probability and Statistical Inference*, Springer-Verlag, New York.

Kantardzic, M. (2003). *Data Mining - Concepts, Models, Methods, and Algorithms*, John Wiley & Sons, Hoboken, NJ, USA.

Kerber, R. (1992). ChiMerge: discretization of numeric attributes. *Proc. of the 10th National Conf. on Artificial Intelligence*, San Jose, CA, pp. 123-128.

Klawonn, F. and Rehm, F. (2006). Clustering Techniques for Outlier Detection. In: *Encyclopedia of Data Warehousing and Mining* (Brennan, E., Bubnis, A., Davies, R. and VanderHook, S., (Eds.)), Hershey, PA, Idea Group Publishing.

Kohavi, R. (1995a). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence*, Montreal, Canada, pp. 1137-1143.

Kohavi, R. (1995b). *Wrappers for Performance Enhancements and Oblivious Decision Graphs*. Ph.D. Thesis, Stanford University, Palo Alto, CA.

Kotsiantis, S. and Kanellopoulos, D. (2006). Discretization Techniques: A recent survey. *GESTS Int. Transactions on Computer Science and Engineering*, Vol. 32, pp. 47-58.

Lee, C. (1994). Generating Classification Rules from Databases. *Proc. of the 9th Conf. on Application of Artificial Intelligence in Engineering*, PA, USA, pp. 205-212.

Li, W., Han, J. and Pei, J. (2001). CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. *Proc. of the Int. Conf. on Data Mining*, San Jose, California, USA, pp. 369-376.

Li, X.-B. (2005). A scalable decision tree system and its application in pattern recognition and intrusion detection. *Decision Support Systems*, Vol. 41, pp. 112-130.

Liu, B., Hsu, W. and Ma, Y. (1998). Integrating Classification and Association Rule Mining. *Knowledge Discovery and Data Mining*, Vol. 98, pp. 80-86.

Liu, H., Hussain, F., Tan, C. L. and Dash, M. (2002). Discretization: An Enabling Technique. *Data Mining and Knowledge Discovery*, Vol. 6, pp. 393-423.

Maimon, O. and Rokach, L. (2007). *Soft Computing for Knowledge Discovery and Data Mining*, Springer-Verlag Inc., New York.

Mehta, M., Agrawal, R. and Rissanen, J. (1996). SLIQ: A Fast Scalable Classifier for Data Mining. *Proc. of the 5th Int. Conf. on Extending Database Technology: Advances in Database Technology*, pp. 18-32.

Mehta, M., Rissanen, J. and Agrawal, R. (1995). MDL-based Decision Tree Pruning. *Proc. of the 1st Int. Conf. on Knowledge Discovery in Databases and Data Mining*, pp. 216-221.

Michalski, R. S. (1969). On the Quasi-Minimal Solution of the General Covering Problem. *Proc. of the 5th Int. Symposium on Information Processing*, Bled, Yugoslavia, pp. 125-128.

Michalski, R. S. and Larson, J. (1975). AQVAL/1 (AQ7) User's Guide and Program Description. *Report No. 731*, Department of Computer Science, University of Illinois, Urbana.

Michalski, R. S., Mozetic, I., Hong, J. and Lavrac, N. (1986). The Multi-Purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains. *American Association of Artificial Intelligence*, Los Altos, CA, pp. 1041-1045.

Mingers, J. (1989). An Empirical Comparison of Pruning Methods for Decision Tree Induction. *Machine Learning*, Vol. 4, pp. 227-243.

Mitchell, T. M. (1997). *Machine Learning*, McGraw-Hill, Columbus, OH, USA.

Mittal, A. and Cheong, L. F. (2002). Employing Discrete Bayes Error rate for discretization and feature selection. *Proc. of the 2002 IEEE Int. Conf. on Data Mining*, pp. 298-305.

Muhlenbach, F. and Rakotomalala, R. (2005). Discretization of Continuous Attributes. In: *Encyclopedia of Data Warehousing and Mining* (Brennan, E., Bubnis, A., Davies, R. and VanderHook, S., (Eds.)), Hershey, PA, Idea Group Publishing.

Niedermayer, D. (2008). An Introduction to Bayesian Networks and their Contemporary Applications. In: *Innovations in Bayesian Networks: Theory and Applications* (Holmes, D. E. and Jain, L. C., (Eds.)), Berlin Heidelberg, Springer-Verlag.

Pagallo, G. and Haussler, D. (1990). Boolean Feature Discovery in Empirical Learning. *Machine Learning*, Vol. 5, pp. 71-99.

Pan, F., Cong, G., Tung, A., Yang, J. and Zaki, M. (2003). CARPENTER: Finding closed patterns in long biological datasets. *Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 637-642.

Pasquier, N., Bastide, Y., Taouil, R. and Lakhal, L. (1999). Discovering Frequent Closed Itemsets for Association Rules. *Lecture Notes in Computer Science*, Vol. 1540, pp. 398-416.

Pei, J., Han, J. and Mao, R. (2000). CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Dallas, TX, USA, pp. 21-30.

Pfahringer, B. (1997). Compression-Based Pruning of Decision Lists. *Proc. of the 9th European Conf. on Machine Learning*, pp. 199-212.

Pham, D. T. and Afify, A. A. (2005a). Online discretization of continuous-valued attributes in rule induction. *Proc. of the Inst. of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Vol. 219, pp. 829-842.

Pham, D. T. and Afify, A. A. (2005b). RULES-6: A Simple Rule Induction Algorithm for Supporting Decision Making. *Industrial Electronics Society, IECON 31st Annual Conf. of IEEE*, Raleigh, North Carolina, USA, pp. 2184-2189.

Pham, D. T. and Afify, A. A. (2006a). SRI: A Scalable Rule Induction Algorithm. *Proc. of the Inst. of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Vol. 220, pp. 537-552.

Pham, D. T. and Afify, A. A. (2006b). Three New MDL-Based Pruning Techniques for Robust Rule Induction. *Proc. of the Inst. of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Raleigh, North Carolina, USA, pp. 553-564.

Pham, D. T. and Afify, A. A. (2007). Clustering Techniques and Their Applications in Engineering. *Proc. of the Inst. of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Vol. 221, pp. 1445-1460.

Pham, D. T. and Aksoy, M. S. (1993). An algorithm for automatic rule induction. *Artificial Intelligence in Engineering*, Vol. 8, pp. 227-282.

Pham, D. T. and Aksoy, M. S. (1995a). A new algorithm for inductive learning. *Journal of Systems Engineering*, Vol. 5, pp. 115-122.

Pham, D. T. and Aksoy, M. S. (1995b). RULES: A Simple Rule Extraction System. *Expert Systems with Applications*, Vol. 8, pp. 59-65.

Pham, D. T., Bigot, S. and Dimov, S. S. (2004). A rule merging technique for handling noise in inductive learning. *Proc. of the Inst. of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Raleigh, North Carolina, USA, pp. 1255-1268.

Pham, D. T. and Dimov, S. S. (1997a). An Algorithm for Incremental Inductive Learning. *Proc. of the Inst. of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Vol. 211, pp. 239-249.

Pham, D. T. and Dimov, S. S. (1997b). An Efficient Algorithm for Automatic Knowledge Acquisition. *Pattern Recognition*, Vol. 30, pp. 1137-1143.

Pham, D. T., Dimov, S. S. and Bigot, S. (2003). RULES-5: A Rule Induction Algorithm for Problems Involving Continuous Attributes. *Proc. of the Inst. of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Vol. 217, pp. 1273-1286.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In: *Machine Learning: An Artificial Intelligence Approach* (Michalski, S. R., Carbonell, G. J. and Mitchell, M. T., (Eds.)), Palo Alto, CA, Tioga Publishing Co.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, Vol. 1, pp. 81-106.

Quinlan, J. R. (1990). Learning Logical Definitions from Relations. *Machine Learning*, Vol. 5, pp. 239-266.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rastogi, R. and Shim, K. (1998). PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning. *Proc. of the 24th Int. Conf. on Very Large Data Bases*, New York, USA, pp. 404-415.

Richeldi, M. and Rossotto, M. (1995). Class-Driven Statistical Discretization of Continuous Attributes. *Proc. of the 8th European Conf. on Machine Learning*, pp. 335-338.

Rissanen, J. (1983). A Universal Prior for Integers and Estimation by Minimum Description Length. *Annals of Statistics*, Vol. 11, pp. 416-431.

Rissanen, J. (1986). Stochastic complexity and modelling. *Annals of Statistics*, Vol. 14, pp. 1080-1100.

Rivest, R. L. (1987). Learning Decision Lists. *Machine Learning*, Vol. 2, pp. 229-246.

RuleQuest (2001) Data Mining Tools C5.0. IN (Ed.^(Eds.) *RuleQuest Research Pty Ltd, 30 Athena Avenue.* ed. Australia, St Ives NSW 2075.

Savasere, A., Omiecinski, E. and Navathe, S. B. (1995). An Efficient Algorithm for Mining Association Rules in Large Databases. *Proc. of the 21st Int. Conf. on Very Large Data Bases*, Zurich, Swizerland, pp. 432-444.

Shafer, J. C., Agrawal, R. and Mehta, M. (1996). SPRINT: A Scalable Parallel Classifier for Data Mining. *Proc. of the 22nd Int. Conf. on Very Large Data Bases*, pp. 544-555.

Shannon, C. E. and Weaver, W. (1963). *A Mathematical Theory of Communication*, University of Illinois Press, Urbana, Illinois.

Thornton, C. J. (1992). *Techniques in computational learning*, Chapman and Hall Computing, London.

Veaux, R. D. (2007) Datasets for use in the Data Mining Course. IN (Ed.^(Eds.) *Department of Mathematics and Statistics, Bronfman Science Center.* ed. MA, 01267, USA, Williams College, Williamstown.

Wang, J., Han, J. and Pei, J. (2003). CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. *Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 236-245.

Wang, J. and Karypis, G. (2006). On Mining Instance-Centric Classification Rules. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 18, pp. 1497-1511.

Wilson, D. R. and Martinez, T. R. (2000). Reduction Techniques for Instance-Based Learning Algorithms. *Machine Learning*, Vol. 38, pp. 257-286.

Witten, I. H. and Frank, E. (2005). *Data Mining - Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Publishers, San Francisco, USA.

Wojtusiak, J., Michalski, R. S., Kaufman, K. A. and Pietrzykowski, J. (2006). The AQ21 Natural Induction Program for Pattern Discovery: Initial Version and its Novel Features. *18th IEEE International Conference on Tools with Artificial Intelligence*, pp. 523-526.

Wong, A. K. C. and Chiu, D. K. Y. (1987). Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 9, pp. 796-805.

Yan, X. and Han, J. (2003). CloseGraph: mining closed frequent graph patterns. *Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Washington, D.C., pp. 286-295.

Yan, X., Han, J. and Afshar, R. (2003). CloSpan: Mining Closed Sequential Patterns in Large Datasets. *Proc. of the 2003 SIAM Int.Conf. on Data Mining*, pp. 166-177.

Yang, Y. and Webb, G. I. (2005). Discretization for Data Mining. In: *Encyclopedia of Data Warehousing and Mining* (Brennan, E., Bubnis, A., Davies, R. and VanderHook, S., (Eds.)), Hershey, PA, Idea Group Publishing.

Yin, X. and Han, J. (2003). CPAR: Classification based on Predictive Association Rules. *Proc. of the 2003 SIAM Int. Conf. on Data Mining*, Cathedral Hill Hotel, San Francisco, CA, pp. 345-360.

Zaki, M. J. and Hsiao, C. J. (2002). CHARM: An Efficient Algorithm for Closed Itemset Mining. *Proc. of the 2002 SIAM Int. Conf. on Data Mining*, pp. 33-43.

Zaki, M. J., Parthasarathy, S., Ogihara, M. and Li, W. (1997). New Algorithms for Fast Discovery of Association Rules. *Third Int. Conf. on Knowledge Discovery and Data Mining*, Newport Beach, California, USA, pp. 283-286.