COST-SENSITIVE DECISION TREE LEARNING USING A MULTI-ARMED BANDIT FRAMEWORK

Susan Elaine LOMAX

SCHOOL OF COMPUTING, SCIENCE AND ENGINEERING,
INFORMATICS RESEARCH CENTRE, COLLEGE OF SCIENCE AND TECHNOLOGY, UNIVERSITY OF SALFORD

Submitted in Fulfilment of the Requirements of the Degree of Doctor of Philosophy, June 2013

CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

ACKNOWLEDGEMENTS

Thanks to John Bacon, Mathscope, University of Salford for help in the development of the solution to calculating potential unique bandit paths and to my co-supervisor Dr Chris Bryant for pointing me in the right direction. Thanks to Dr K J Abrams for her help in understanding the Annealing dataset along with advice regarding allocation of test costs and groupings of attributes. Thanks also to Duncan Botterill, without whose help the experimental results would have taken even longer to process.

The majority of my thanks go to my supervisor Professor Sunil Vadera for his continuing help and support whilst undertaking this research. He has given me the confidence to believe in myself when otherwise I would not.

As part of this research the following papers have been published which contain some of the content of this thesis:

An Empirical Comparison of Cost-Sensitive Decision Tree Induction Algorithms published in Expert Systems: The Journal of Knowledge Engineering, July, Vol. 28, No 3, 227 – 268.

A Survey of Cost-Sensitive Decision Tree Induction Algorithms published in ACM Computing Surveys, Vol. 45, No. 2, Article 16.

A Multi-Armed Bandit Approach to Cost-Sensitive Decision Tree Learning published in Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference, 10-13 December 2012, Brussels, Belgium, 162 – 168.

Acknowledgement also goes to the reviewers and editors of the above publications for their useful comments which led to improvements to the content and presentation of the publications and ultimately this thesis.

ABSTRACT

Decision tree learning is one of the main methods of learning from data. It has been applied to a variety of different domains over the past three decades. In the real world, accuracy is not enough; there are costs involved, those of obtaining the data and those when classification errors occur. A comprehensive survey of cost-sensitive decision tree learning has identified over 50 algorithms, developing a taxonomy in order to classify the algorithms by the way in which cost has been incorporated, and a recent comparison shows that many cost-sensitive algorithms can process balanced, two class datasets well, but produce lower accuracy rates in order to achieve lower costs when the dataset is less balanced or has multiple classes.

This thesis develops a new framework and algorithm concentrating on the view that cost-sensitive decision tree learning involves a trade-off between costs and accuracy. Decisions arising from these two viewpoints can often be incompatible resulting in the reduction of the accuracy rates.

The new framework builds on a specific Game Theory problem known as the multi-armed bandit. This problem concerns a scenario whereby exploration and exploitation are required to solve it. For example, a player in a casino has to decide which slot machine (bandit) from a selection of slot machines is likely to pay out the most. Game Theory proposes a solution of this problem which is solved by a process of exploration and exploitation in which reward is maximized. This thesis utilizes these concepts from the multi-armed bandit game to develop a new algorithm by viewing the rewards as a reduction in costs, utilizing the exploration and exploitation techniques so that a compromise between decisions based on accuracy and decisions based on costs can be found. The algorithm employs the adapted multi-armed bandit game to select the attributes during decision tree induction, using a look-ahead

methodology to explore potential attributes and exploit the attributes which maximizes the reward.

The new algorithm is evaluated on fifteen datasets and compared to six well-known algorithms J48, EG2, MetaCost, AdaCostM1, ICET and ACT. The results obtained show that the new multi-armed based algorithm can produce more cost-effective trees without compromising accuracy. The thesis also includes a critical appraisal of the limitations of the developed algorithm and proposes avenues for further research.

CHAPTER 1: INTRODUCTION

1.1 The motivation for the research in this thesis

Decision trees are a natural way of presenting a decision-making process, because they are simple and easy for anyone to understand (Quinlan 1986). Learning decision trees from data however is more complex, with most methods based on an algorithm known as ID3 which was developed by Quinlan (1979, 1983, 1986). ID3 takes a table of examples as input, where each example consists of a collection of attributes, together with an outcome (or class) and induces a decision tree, where each node is a test on an attribute, each branch is the outcome of that test and at the end are leaf nodes indicating the class to which an example, when following that path, belongs. ID3, and a number of its immediate descendents, such as C4.5 (Quinlan 1993), CART (Breiman et al. 1984) and OC1 (Murthy et al. 1994) focused on inducing decision trees that maximized accuracy.

However, several authors have recognized that in practice there are costs involved (e.g. Breimen et al. (1984); Turney (1995, 2000); Elkan (2001)). For example, it costs time and money for blood tests to be carried out (Quinlan et al. 1987). In addition, when examples are misclassified, they may incur varying costs of misclassification depending on whether they are false negatives (classifying a positive example as negative) or false positives (classifying a negative example as positive). This has led to many studies which develop algorithms that aim to induce cost-sensitive decision trees.

Lomax and Vadera (2013) present a survey of cost-sensitive decision tree algorithms. Some past comparisons by Vadera and Ventura (2001) and a more comprehensive evaluation by

Lomax and Vadera (2011) have evaluated algorithms which have incorporated these costs into the construction using extensions to statistical measures, genetic algorithms, or boosting and bagging techniques. Experiments were carried out over a range of cost matrices and showed that using both costs in the construction was the better method. However, on a later examination of the results from the algorithm which performed better overall (ICET introduced by Turney (1995)), variations in performance revealed weaknesses where the algorithm produced poorer results than would be expected. Contributing factors to the variation in performances are thought to be large numbers of attributes, differing numbers of attribute values, class distribution, number of classes and differing costs. Trade-off between high misclassification costs result in the sacrifice of the accuracy rate. The nature of the dataset may account for some of the discrepancies. These could influence how easy it is for the algorithm to classify examples.

Although in the literature, it has always been suggested that Game Theory is different to decision theory (an idea utilized in some decision tree algorithms) and therefore caters to different decision situations, it has been suggested that it can be applied in a Machine Learning capacity (Cesa-Bianchi and Lugosi 2006) and could be used for prediction as both disciplines have in common the idea that past experience predicts future events.

Machine Learning techniques are used in predictions with cost-sensitive learning as a descendent of this technique. Game Theory can also be used to predict outcomes by choosing strategies according to and linked with ‘payoffs’. The pay-offs vary but can easily be described as ‘costs’. For example in cost-sensitive learning the goal is to reduce costs, therefore the pay-off is simply the reduction of cost or to obtain the lowest cost as possible.

Pay-off functions are assigned to the strategies in order to help make the decisions. Picking strategies which would maximize pay-off is the desired outcome with a trend towards simplicity. Finding the simplest assumption needed (Occam's razor[1]) is the ideal outcome (Rasmusen 2001). Pay-offs are shown using a matrix and strategies can be illustrated using decision-tree like structures.

Use of costs within the decision tree learning process has introduced many interesting problems involving the trade-off required between accuracy and costs. It is clear that, whilst there are existing cost-sensitive decision tree algorithms which can solve two-class balanced problems well, other types of problems cause difficulties. In particular several authors have recognized that there can be a trade-off between accuracy and minimizing cost (Lomax and Vadera 2009, 2011) or a reduction in performance (Ting 2000a).

Hence, this research aims to utilize Game Theory as a basis for developing a cost-sensitive decision tree algorithm, which aims to be able to address the trade-off between accuracy and cost that has been observed in previous studies.

1.2 Research methodology

Different methodologies have been studied and the most appropriate one is selected for this PhD. The methodologies studied are grouped under three categories:

[1] The simplest solution is often the most likely one.

- Constructive Methods

  These deal with conceptual and technical development and may not be as empirically based as other methods. They may involve evaluating prototype software against defined criteria or testing prototypes (Livari et al. 1998).

- Nomothetic Methods

  These are positivism, where the idea is that the world exists externally and measurements should be through objective methods. It encourages statistics and experiments (Livari et al. 1998). Methods in this category are generally 'laws' i.e., laws of physics etc, and scientific methods such as hypothesis testing, mathematical analysis, experiments, field studies and surveys. They will be quantitative in the data collection and confirmatory.

- Idiographic Methods

  These are interpretivism, which is the opposite of positivism. It encourages the appreciation of constructions and meanings which people have, not reporting facts, but dealing with the interpretations of them (Livari et al. 1998). Methods in this category are generally case studies and action research, dealing with direct experience. Case studies and action research deal in on-the-spot fact finding. The researcher learns about system requirements or how things work by visiting the place which requires it or has information about it. They are useful for studying how and why things happen. They are performed by interviews, observations, which are either overt or covert, and document analysis.

Comparing the different methodologies listed above, a method belonging to the second category, Nomothetic methods, is more appropriate for this research as these methods involve hypothesis testing and experimentation as described by Livari et al (1998) p.187.

A method named GQM (Goal, Question, Metric) (Basili and Weiss 1984) is recommended to be used in the area of software development. GQM is a measurement mechanism used in order to obtain feedback and evaluation in software development. Its aim is to focus on specific goals and must be defined in a top-down fashion. It is ideally suited to this area as there are many observable characteristics such as lines of code, number of defects or complexity, making other methods which are metric-driven and bottom up unworkable (van Solingen et al. 2002). The GQM approach is to define goals, which are then refined into questions, and metrics are used in order to gain enough information so that the questions can be answered.

In this thesis, the goal is to develop a framework which uses the trade-off required between accuracy and costs in order to achieve low costs and high accuracy which is required in cost-sensitive learning. As a result of this goal, the following questions have been developed:

1. How well do existing cost-sensitive decision tree algorithms perform?
2. What are the weaknesses of existing cost-sensitive decision tree algorithms?
3. Is it possible to minimize costs and minimize the sacrifice of the accuracy rate which occurs in cost-sensitive decision tree learning?
4. Will using a technique, which has been developed to deal with trade-off by using pay-offs, help in achieving the aim of cost-sensitive decision tree learning?

In order to determine the answers to these questions, metrics have been devised which measure cost and determine whether these costs are minimized and accuracy, determining whether this is maximized or that the sacrifice is minimized. A research hypothesis has been developed with the aim of showing what happens to the accuracy and costs when using the trade-off effectively.

1.3 Research hypothesis, aims and objectives

The Research Hypothesis put forward is that cost-sensitive decision tree learning involves a trade-off between decisions based on accuracy and decisions based on costs and that Game Theory can be utilized to develop an algorithm that improves upon the performance of existing algorithms. By using Game Theory, it may be possible to explore the opposing decisions in such a way as to decide that a compromise can indeed be reached. Any algorithm which aims to be a cost-sensitive one will need to achieve this trade-off in order to function correctly. The aim of this PhD is therefore to show what happens to accuracy and costs in the trade-off and to find a framework that can use the trade-off effectively to achieve the low costs and high accuracy required in cost-sensitive learning. In order to test this hypothesis the thesis objectives are:

1. To survey and review existing cost-sensitive decision tree algorithms in order to investigate ways in which costs have been introduced into the decision tree learning process and at which stages they have been introduced
2. To evaluate existing cost-sensitive decision tree algorithms in order to discover whether these algorithms are successful over many types of problems or are only effective for some types of problems, for example binary class datasets or balanced datasets
3. To develop a new cost-sensitive decision tree algorithm which is based on Game Theory
4. To investigate and evaluate the new algorithm against existing algorithms and measure performance in terms of cost to classify and accuracy, in order to test the research hypothesis

1.4 Outline of thesis

The rest of the thesis is structured as follows:

- Chapter 2: Background

  This chapter presents the background to decision tree learning and Game Theory

- Chapter 3: Survey of existing cost-sensitive decision tree algorithms

  This chapter presents the results of a literature search which identifies existing cost-sensitive decision tree algorithms and categorizes them into classes by the way the costs have been introduced

- Chapter 4: The development of a new multi-armed bandit framework for cost-sensitive decision tree learning

  This chapter presents an analysis of previous cost-sensitive decision tree algorithms, highlights their weaknesses and suggests a new framework using multi-armed bandits. Experiments are carried out in order to fine-tune the algorithm and an extension is also developed. The experimental methodology is also presented

- Chapter 5: Investigating parameter settings for MA_CSDT

  This chapter presents an extensive investigation into four areas. These four areas address the parameter settings, in particular those which determine whether it is worthwhile to continue the induction process, determining the number of lever pulls to set for a dataset, which version and strategy is better and to investigate how different combinations of parameter settings and strategies can be used to obtain good results. Guidelines to setting these parameters are also discussed

- Chapter 6: An empirical comparison of the new algorithm with existing cost-sensitive decision tree algorithms

This chapter presents the results of the empirical comparison and evaluation against existing cost-sensitive decision tree algorithms and an accuracy-based algorithm in order to determine whether the aim of the algorithm can be met

- Chapter 7: Conclusions and future work

  This chapter summarizes the aims and objectives of this thesis and discusses the results obtained both from the investigation and the empirical comparison. It gives details of work which could be carried out in the next stage of the algorithm's development and suggests other possible future experiments based on the findings of this thesis

CHAPTER 2: BACKGROUND

2.1 Decision tree learning

Given a set of examples, early decision tree algorithms, such as ID3 and CART, utilize a greedy top-down procedure. An attribute is first selected as the root node using a statistical measure (Quinlan 1979, 1983; Breiman et al. 1984). The examples are then filtered into subsets according to values of the selected attribute. The same process is then applied recursively to each of the subsets until a stopping condition, such as all of the examples in the subset being of the same class. The leaf nodes are then assigned the majority class as the outcome. Researchers have experimented with different selection measures, such as the GINI index (Breiman et al. 1984), using chi-squared (Hart 1985) and which have been evaluated empirically (Mingers 1989). The selection measure utilized in ID3 is based on Information Theory which provides a measure of disorder, often referred to as the entropy, and which is used to define the expected entropy, E for an attribute A (Shannon 1948; Quinlan 1979; Winston 1993). The entropy of an attribute A is defined as:

$$E(A) = \sum_{a \in A} P(a). \sum_{c \in C} -\mathrm{P}(a|c)\log 2(\mathrm{P}(a|c))) \qquad (2.1)$$

where $a \in$ A are the values of attribute A, and the $c \in$ C are the class values.

This formula measures the extent to which the data is homogeneous. For example, if all the data were to belong to the same class, the entropy would be '0'. Likewise if all the examples belonged to different classes, the entropy would be '1'. ID3 uses an extension of the entropy by calculating the gain in information *(I)* achieved by each of the attributes if they were chosen for the split and choosing the attribute which maximizes this gain. This is given by equation (2.2).

ID3: $$I_A = E(D) - E(A) \quad (2.2)$$

where $E(D) = \sum_{c \in C} -\frac{N_c}{N} log_2 \frac{N_c}{N}$, calculated on the current training set before splitting.

Although Quinlan adopted this measure for ID3, he noticed that the measure is biased towards attributes which have more values, and hence proposed a normalization, known as the Gain Ratio, which is defined by:

C4.5: $$GainRatio_A = \frac{I_A}{Info_A} \; where \; Info_A = \sum_{a \in A} -\frac{N_a}{N} log_2 \frac{N_a}{N} \quad (2.3)$$

C4.5 was also developed to include the ability to process numerical data and deal with missing values. Figure 1 presents the tree that result from applying the ID3 procedure to the examples in Table 1. At each leaf is the class distribution, in the format of (faulty, not faulty).

picture quality	sound quality	age	class
poor	good	2	faulty
poor	excellent	1	faulty
good	poor	2	faulty
good	poor	2	faulty
good	excellent	1	not faulty
good	good	1	not faulty
good	good	2	faulty
excellent	good	1	faulty
excellent	excellent	1	not faulty
excellent	good	2	not faulty
good	good	2	faulty
good	good	2	faulty
good	good	1	not faulty
excellent	excellent	1	not faulty
excellent	good	1	not faulty

Table 1 Example dataset 'Television Repair'

Once a decision tree has been built, some type of pruning is then usually carried out. Pruning is the term given to that of replacing one or more sub-trees with leaf nodes. There are three main reasons for pruning. One is that it helps to reduce the complexity of a decision tree, which would otherwise make it very difficult to understand (Quinlan 1987), resulting in a faster, possibly less costly classification. Another reason is to help prevent the problem of over-fitting the data.

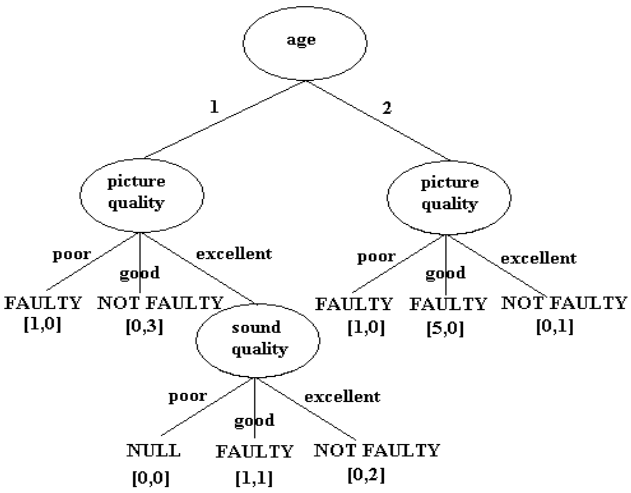


Figure 1 Decision tree after ID3 has been applied to the dataset in Table 1

The third reason is that noisy, sparse or incomplete datasets can cause very complex decision trees, so pruning is a good way to simplify them (Quinlan 1987). There are several ways to calculate whether a sub-tree should be pruned or not. Quinlan (1987), Knoll et al. (1994) and Bradford et al. (1998a, 1998b) have discussed different methods to do this, for instance, aiming to minimize loss (Bradford et al. 1998a, 1998b), or using misclassification costs to prune a decision tree (Knoll et al. 1994). A comprehensive review on pruning methods has been carried out by Frank and Witten (1998).

2.2 Game Theory

Game Theory is a discipline which deals in trade-off. It deals with types of decision making where there may be more than one decision-maker. Game Theory (Davis 1983; Osborne 2004) states that it is decision making with two decision-makers, which it denotes as 'players'. At least two players choose a strategy (make a decision) and as a result a reward or pay-off occurs. Each player must worry about what the other is doing. Pseudo-players have actions taken in a mechanical way; Nature is an example of a Pseudo-player (Rasmusen 2001). Game Theory is the theory of many games not just one (Davis 1983).

Game Theory differs from other types of decision making problems because as decision-makers are manipulating the environment i.e. deciding how much advertising space to purchase, the environment i.e. other decision-makers are trying to do the same (Davis 1983). Game Theory has been used in disciplines such as economics, social science, political science and biology and applied to tasks such as price fixing, advertising, and strategies used in competitive business.

Game Theory aims to help understand situations where decision-makers interact with each other according to a set of rules and consists of a collection of models which need to be simple with assumptions capturing the essence of the situation (Osborne 2004). Many problems can be understood without special technical background (Davis 1983). Real games are very complicated and toy games are often used instead (Binmore 2007). Applications which can be reduced to a single problem, for example a shop keeper reducing prices of his stock in response to a competitor doing likewise, are all situations for which Game Theory can be applied.

Decisions are linked to goals and the consequences of each option must be known in order to make the solution easy. The best strategy is chosen in order to reach the goal. If chance plays a role, decisions are harder to make (Davis 1983). Pay-off functions are assigned to strategies in order to help make the decisions. Picking strategies which maximizes pay-off is the desired outcome with a trend towards simplicity; finding the simplest assumption needed is the ideal outcome (Rasmusen 2001). Pay-offs are shown using a matrix and strategies can be illustrated using decision-tree like structures.

Models are not either right or wrong but useful or not depending on the purpose for which they are used. The models are examined in order to analyze their implications, to either confirm an idea or suggest it is wrong. This analysis should help understand why it is wrong. Time is absent from the model. Each player chooses their actions "simultaneously" in that no player is informed when an action is chosen or what action another player has chosen (Osborne 2004). The assumption is that actions are chosen once and for all. It is assumed that all players will try to do their best. A Nash Equilibrium is a pair of strategies which, when applied, results in both players choosing the same option as neither wishes a change in strategy. It occurs when all players make the best reply to the strategy choice of the others (Nash 1950a, 1950b). For example if a player knew that the other player would always choose a particular strategy, they could maximize their pay-off by choosing the same strategy.

There are three main categories of games in Game Theory. These are (i) the two-person zero-sum game (ii) the two-person non-zero-sum game and (iii) the *n*-person game first defined by von Neumann and Morgenstern (1953) which involves forming coalitions. In a zero-sum game one person's gain is the other's loss whereas in a non-zero-sum person game this is not

the case. An example of a zero-sum game would be Matching Pennies where each player has two strategies; heads or tails. The Prisoner's Dilemma is an example of a non-zero-sum person game.

The Prisoner's Dilemma Game is one of the most well known 'games' in Game Theory (Binmore 2007). Two suspects have been arrested for a minor crime, for instance handling stolen goods, for which there is ample evidence. However, they are suspected by the police of the greater crime of burglary for which there is only circumstantial evidence and no proof. The suspects are both offered the same deal:

- If one confesses and turns Queen's evidence, and the other does not, he will go free and the other goes to jail for a maximum prison term.
- If both suspects confess they both go to jail for a minimum prison term for burglary.
- If both suspects remain silent they both go to jail for a year for the handling charge as there is no evidence for any other wrong-doing.

The assumption is that each prisoner will try their upmost to do what is best for themselves. The pay-offs are displayed in a matrix presented in Table 2.

		suspect 2	
		confess	do not confess
suspect1	confess	min, min	0, max
	do not confess	max, 0	1,1

Table 2 Pay-off matrix for Prisoner's Dilemma

Each player has two basic choices; they can act co-operatively or un-cooperatively. For any fixed strategy of the other players, a player always does better by playing un-cooperatively than by playing co-operatively (Binmore 2007).

Another well-known game is the Hawk-Dove Game, where two birds, for example pheasants, may contest a resource such as food. The two birds can either act passively or aggressively in this kind of situation (Binmore 2007). A passive bird would surrender the food to an aggressive bird. Two passive birds would share the food but two aggressive birds would fight. Passive birds are usually referred to as 'doves' and aggressive birds as 'hawks' (Maynard Smith 1984). Each prefers to be aggressive if the other is passive and passive if the other is aggressive (Osborne 2004). Pay-off values here which may identify the Hawk-Dove Game with the Prisoner's Dilemma Game are not realistic as injury to either bird would be a serious handicap (Binmore 2007).

An example of the application of Game Theory is in the advertising sector (Davis 1983). Suppose that there are two companies which make a similar product, for example washing powder[2]. The first company A has enough money set aside to buy two blocks of television advertising time and the other B three blocks of time. Each block is one hour long.

The television company splits the day into three time periods; morning (m), afternoon (a) and evening (e). The purchasing of the advertising slots must be made in advance and are confidential. Statistics provided by the television company state that 50% of the audience watches TV in the evening, 30% in the afternoon and 20% in the morning. It is assumed in this example that no-one watches more than one period in a day. If a company buys more

[2] This illustration of an application courtesy of Davis (1983)

time during any time period than the other company it will capture the entire audience during that period, if both companies buy the same number of hours during any one period or neither company buys any time at all during any one period, each get half the audience.

If each member of the TV audience buys the product of just one of the companies, how then should the company allocate their TV time and what percent of the market might they get? There are 6 strategies for the company buying two blocks and 10 for the company buying three blocks. One solution would be that:

- Company B plays each of the strategies (e,e,e), (e,e,a) and (e,a,m) where each letter is a time slot representing one of the blocks of time this company will have purchased, one third of the time
- Company A plays each of the strategies (e,e) $6/15^{th}$ of the time, (a,a,) $5/15^{th}$ of the time and (a,m) $4/15^{th}$ of the time.

If Company B uses these recommended strategies it can be sure of winning on average 63.33% of the time and if Company A uses its recommended strategy, Company B will not win any more than this (Davis 1983).

The Multi-Armed Bandit Game, first proposed by Robbins (1952), is a scenario where a gambler must choose which slot machine from a selection of slot machines to play. He pulls the lever of one of the machines and receives a payoff. The gambler's purpose is to maximize his return i.e. the sum of the pay-offs obtained over a random number of lever pulls. There is a trade-off here between exploration and exploitation as, if the gambler plays only one machine which he thinks is best he may miss out on another machine about to pay out. On the

other hand too much time spent trying out all the slot machines may not actually return a high enough reward (Auer et al. 2001, 2003).

Cost-sensitive decision tree learning involves building a model in a cost-effective way. Other games do not offer this ability as they are not likely to be able to be adapted to include models. Based on the needs of cost-sensitive decision trees in that models need to be induced, the Multi-Armed Bandit Game looks promising in that its lever pulls could be viewed as generating models, and could be mapped to paths contained in a decision tree model. How this could be achieved is discussed further in Chapter 4.

CHAPTER 3: SURVEY OF EXISTING COST-SENSITIVE DECISION TREE ALGORITHMS

Chapter 2 summarizes the main idea behind decision tree induction algorithms that aim to maximize accuracy. How can we induce decision trees that minimize costs? The survey reveals several different approaches. First, some of the algorithms aim to minimize just costs of misclassification, some aim to minimize just the cost of obtaining the information and others aim to minimize both costs of misclassification as well as costs of obtaining the data. Secondly, the algorithms vary in the approach they adopt. Figure 2 summarizes the main categories that cover all the algorithms found in this survey. There are two major approaches: methods that adopt a greedy approach that aims to induce a single tree, and non-greedy approaches that generate multiple trees. Methods that generate single trees include early algorithms, such as *CS-ID3* (Tan and Schlimmer 1989), that adapt entropy-based selection methods to include costs and post-construction methods such as *AUCSplit* (Ferri et al. 2002) that aim to utilize costs after a tree is constructed. Algorithms that utilize non-greedy methods include those that provide a wrapper around existing accuracy based methods, such as *MetaCost* (Domingos 1999), genetic algorithms, such as *ICET* (Turney 1995), and algorithms that adopt tentative searching methods.

Table 3 categorizes the algorithms identified in the literature with respect to the taxonomy shown in Figure 2 and shows the significant volume of work in this field in each of the classes. The table also indicates whether the algorithms incorporate test costs, misclassification costs or both. The time line of algorithms, shown as Figure 3, is also interesting. The first mention of the importance of costs dates back to Hunt's (1966) Concept Learning System framework (CLS) that aimed to develop decision trees and recognized that tests and misclassifications could have an economic impact on human decision making.

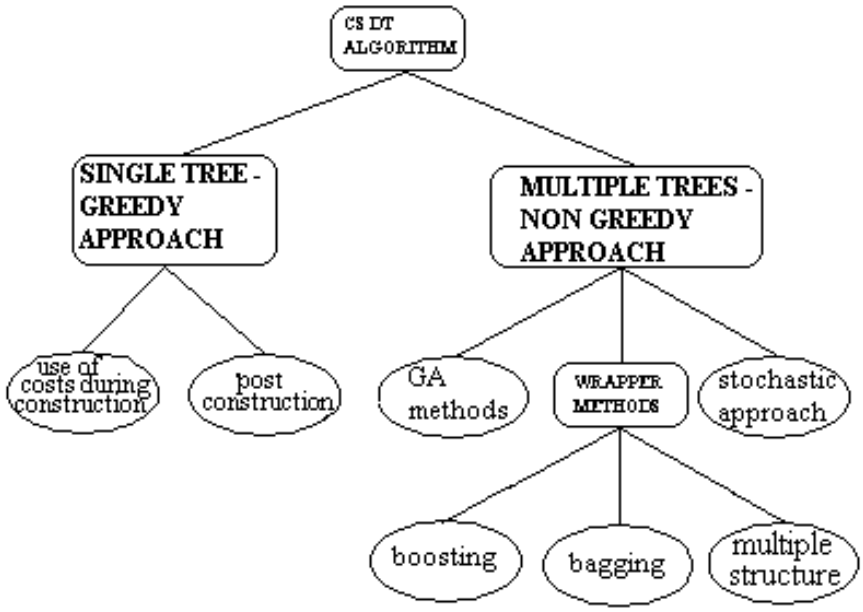


Figure 2 Taxonomy of Cost-Sensitive Decision Tree Induction Algorithms

Although, ID3 adopts some of the ideas of CLS, a significant difference in the development was ID3's use of an information theoretic measure for attribute selection (Quinlan 1979). The use of an information theoretic top-down approach in ID3 influenced much of the early work which focused on methods for adapting existing accuracy based algorithm to take account of costs. These early approaches were evaluated empirically by Pazzani et al. (1994) who observed little difference in performance between algorithms that used cost-based measures and ones that used information gain. This, together with the publication of the results of the *ICET* system (Turney 1995), which used genetic algorithms led to significant interest in developing more novel algorithms, including intense research on the use of boosting and bagging (Ting and Zheng 1998a, 1998b; Ting 2000a, 2000b; Domingos 1999; Zadrozny et al. 2003a, 2003b; Lozano and Abe 2008) and more recently, on the use of stochastic approaches (Esmeir and Markovitch 2007, 2008, 2010, 2011). Table 4 contains notation used in the rest of this chapter.

ALGORITHM	SOURCE
Use of costs during construction	
GINI Altered Priors	Breiman et al. 1984
CS-ID3 *	Tan & Schlimmer 1989, 1990, Tan 1993
IDX *	Norton 1989
EG2 *	Nunez 1991
LMDT	Draper et al. 1994
Cost-Minimization	Pazzani et al. 1994
C4.5CS	Ting 1998, 2002
EvalCount, MaxCost, AvgCost	Margineantu & Dietterich 2003
Decision Tree with Minimal Costs †	Ling et al. 2004
Decision Tree with Minimal Costs Under Resource Constrains †	Qin et al. 2004
DTNB †	Sheng & Ling 2005
CSNL	Vadera 2005a
LDT	Vadera 2005b
Performance †	Ni et al. 2005
CSGain, CSGainRatio *	Davis et al. 2006
CSTree	Ling et al. 2006a
LazyTree †	Ling et al. 2006b
PM †	Liu 2007
CTS-DT †	Zhang et al. 2007
CS-C4.5 *	Freitas et al. 2007
Post construction	
I-gain (Cost-Laplace prob)	Pazzani et al. 1994
AUCSplit	Ferri et al. 2002
GA methods	
ICET †	Turney 1995
ECCO †	Omielan 2005
CGP	Li et al. 2005
GCT_MC	Kretowski & Grzes 2007
Boosting	
UBoost, Cost-UBoost	Ting & Zheng 1998a
AdaCost	Fan et al. 1999, Ting 2000b
CSB1, CSB2	Ting 2000b
SSTBoost	Merler et al. 2003
GBSE, GBSE-T	Abe et al. 2004
JOUS-Boost	Mease et al. 2007
Lp-CSB, Lp-CSB-PA, Lp-CSB-A	Lozano & Abe 2008
Bagging	
MetaCost	Domingos 1999
MetaCost_A, MetaCost_CSB	Ting 2000a
Cost plus Prior Probability, Cost-Only	Lin & McClean 2000
Costing (Black box)	Zadrozny et al. 2003a, 2003b
B-PET, B-LOT	Moret et al. 2006
Multiple Structure	
Multi-tree	Estruch et al. 2002
Stochastic approaches	
ACT †	Esmeir & Markovitch 2007, 2008
TATA†	Esmeir & Markovitch 2010, 2011

Table 3 Cost-sensitive decision tree induction algorithms categorized with respect to taxonomy by time

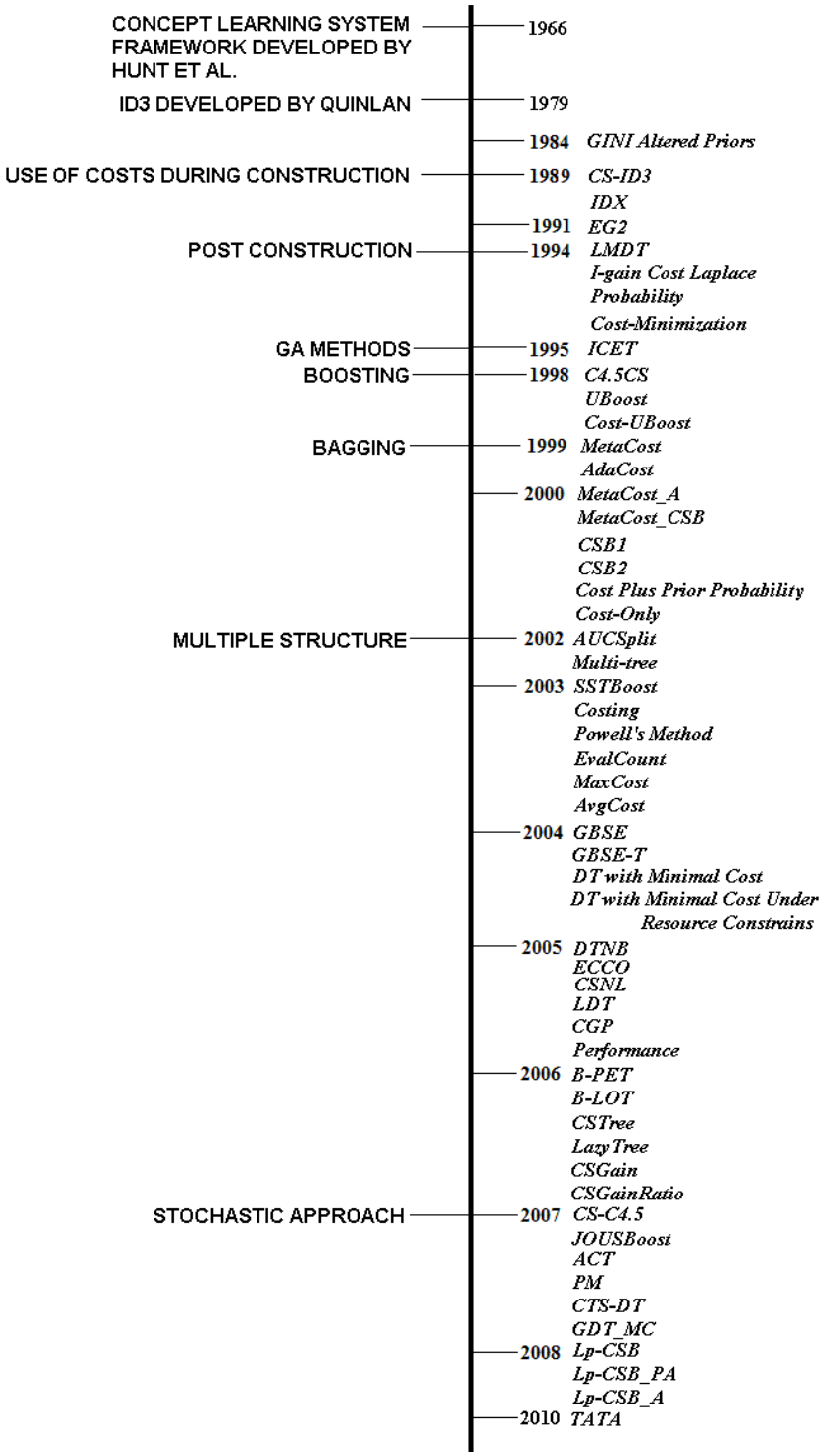


Figure 3 A timeline of algorithms

Symbol	Definition
N	Number of examples in current training set/node
N_i	Number of examples in training set belonging to class *i*
x	Refers to an example in the training set
node(x)	Leaf node to which the example belongs
k	Number of classes and indicates looping through each class in turn
w	Weights
A	Indicates an attribute
a	Indicates attribute values belonging to an attribute
C_{ij}	Misclassification cost of classifying a class *i* example as a class *j* example
C_A	Test cost for attribute A
cost(x,y)	Cost of classifying example *x* into class *y*
h_i	The i^{th} hypothesis

Table 4 Definitions of equations

3.1 Single tree, greedy cost-sensitive decision tree induction algorithms

As described in Chapter 2.1, historically, the earliest tree algorithms developed top-down greedy algorithms for inducing decision trees. The primary advantage of such greedy algorithms is efficiency, though a potential disadvantage is that they may not explore the search space adequately to obtain good results. This section presents a survey of greedy algorithms. The survey identified two major strands of research: Section 3.1.1.1 describes algorithms that utilize costs during tree construction and Section 3.1.2 describes post-construction methods that are useful when costs may change frequently.

3.1.1 Use of costs during construction

3.1.1.1 The extension of statistical measures. As outlined in the previous section, top-down decision tree induction algorithms use a measure, such as information gain, to select an attribute upon which the dataset will be partitioned during the tree induction process. A reasonable extension, which was taken by a number of early algorithms, was to adapt these information theoretic measures by including costs. These early algorithms retained the top-down induction process and the only differences between them are the selection measures and whether they take account of costs of attributes as well as costs of misclassification.

Five of the algorithms, *CS-ID3* (Tan and Schlimmer 1989), *IDX* (Norton 1989), *EG2* (Núnez 1991) , *CSGain* (Davis et al. 2006) and *CS-C4.5* (Freitas et al. 2007) focus on minimizing the cost of attributes and adapt the information theoretic measure to develop a cost based attribute selection measure, called the Information Cost Function for an attribute *A* (ICF_A):

EG2:	$ICF_A = 2^{InfoGainA} - 1/(C_A + 1)^{\omega}$	(3.1)
CS-ID3:	$ICF_A = (InfoGain_A)^2 / C_A$	(3.2)
IDX :	$ICF_A = InfoGain_A / C_A$	(3.3)
CS-C4.5:	$ICF_A = InfoGain_A / (C_A \phi_A)^{\omega}$	(3.4)
CSGain:	$ICF_A = (N_a/N) * InfoGain_A - \omega * C_A$	(3.5)

These measures are broadly similar in that they all include the cost of an attribute (C_A) to bias the measure towards selecting attributes that cost less but still take some account of the information gained. The only difference between the measures is the extent of weight given to the cost of an attribute, with *EG2* and *CS-C4.5* adopting a user provided parameter ω that varies the extent of the bias. *CS-C4.5* also includes ϕ_A, a risk factor used to penalize a particular type of tests, known as delayed tests, which are tests, such as blood tests, where there is a time lag between requesting and receiving the information. The authors of *CSGain* also experiment with a variation, called *CSGainRatio* algorithm where they use the Gain ratio instead of the information gain.

Figure 4 presents a cost-sensitive decision tree induced by applying the *EG2* algorithm to the data in Table 1. For illustration purposes, the attributes picture quality, sound quality and age are assigned random test costs of 30, 15 and 1 units respectively. These costs are used in selecting an attribute using the ICF measure resulting in a tree that takes account of the costs of the tests.

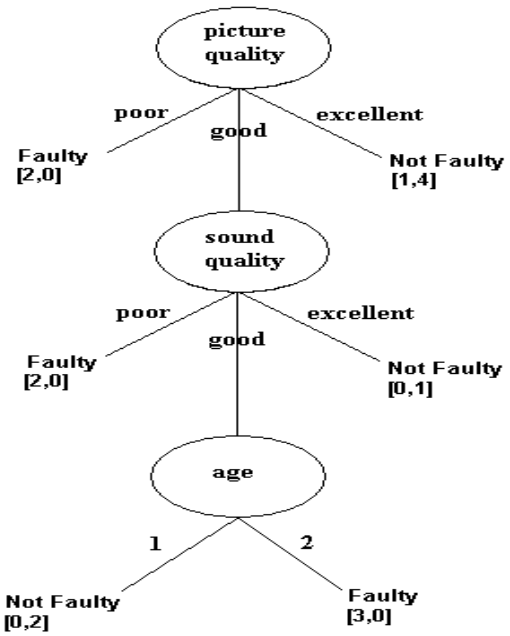


Figure 4 Decision tree after *EG2* has been applied to the dataset in Table 1

Algorithms that continue this adaptation of information theoretic measures but also take account of the misclassification cost as well as the test costs include an approach by Ni et al. (2005), Zhang et al. (2007), Zhang (2010) and Liu (2007). Although the detailed measures differ, they all aim to capture the trade-off between the cost of acquiring the data and its contribution to reducing misclassification cost. Ni et al. (2005), for example, utilize the following attribute selection measure:

$$Performance:\ ICF_A = \left(\left(2^{GainRatio_A} - 1\right) * DMC_A/(C_A + 1)\right)\ *\ \acute{\omega}_A \qquad (3.6)$$

where $\acute{\omega}_A$ is the bias of experts for attribute A and DMC_A is the improvement in misclassification cost if the attribute A is used.

As well as using both types of cost, this algorithm makes use of domain experts who assign a value of importance to each of the attributes. If an expert has no knowledge of the importance

of an attribute this bias is set to the default value of 1. If some attributes produce the same value for equation (3.6), preference is given to those attributes with the largest reduction in misclassification costs (DMC_A). If this fails to find an attribute then the attribute with the largest test cost (C_A) is chosen as the aim is to reduce misclassification costs.

Liu (2007) identifies some weaknesses of equation (3.6), noting that several default values have been used, so develops the *PM* algorithm. Liu (2007) notes that if gain ratios of attributes are small, the values returned by the original algorithm, equation (3.6), would be small; resulting in the costs of attributes being ignored. If attributes have large total costs, the information contained in those attributes will be ignored. Other issues are the conflict of applying resource constrains. For instance, the overall aim of this algorithm is to allow for user resource constrains and it is therefore necessary to allow for the fact that users with increased test resources are not concerned as much about the cost of attributes, rather in the reduction of misclassification costs, and alternatively those with limited test resources are more concerned with the cost of the tests in order to reduce the overall costs rather than only reducing the misclassification costs.

In order to trade off between these needs, a solution offered by Liu (2007) is to normalize the gain ratio values and to employ a harmonic mean to weigh between concerns with test costs (low test resources) and reduction in misclassification costs (when test resources are not an issue), additionally a parameter α is used to balance requirements of different test examples with different test resources.

Zhang et al. (2007) take a different approach when adapting the *Performance* algorithm. They focus on the fact that the test costs and misclassification costs are possibly not on the

same scale; test costs would be considered on a cost scale of currency whilst misclassification costs, particularly in terms of medical diagnosis, states Zhang et al. (2007), must be a social issue; what monetary value could be assigned for potential loss of life? The adaptation attempts to achieve maximal reduction in misclassification costs from lower test costs. The only difference to equation (3.6) to produce *CTS (Cost-Time Sensitive Decision Tree)*, is to remove the bias of expert parameter, preferring to address such issues as waiting costs (also referred to in other studies as delayed cost), at the testing stage by developing appropriate test strategies.

The above measures all utilize the information gain as part of a selection measure. An alternative approach, taken by Breiman et al. (1984), is to alter the class probabilities, $P(_i)$ used in the information gain measure. That is, instead of estimating P(i) by N_i/N, it is weighted by the relative cost, leading to an altered probability (Breiman, et al. 1984, p114):

$$\text{Altered Probability}_i = C_{ij}*(N_i/N) / \sum_j \text{cost}(j)(N_j/N) \quad (3.7)$$

In general, the cost of misclassifying an example of class *j* may also depend on the class *i* that it is classified into, so Breiman et al. (1984) suggest adopting the sum of costs of misclassification:

$$\text{cost}(j) = \sum_i C_{ij} \quad (3.8)$$

Although these altered probabilities can then be used in the Information Gain measure, the method was tried by Pazzani et al. (1994) using the GINI index:

$$\text{Altered GINI} = 1 - \sum_{y=1}^{k} \text{Altered Probability}_y^2 \quad (3.9)$$

C4.5 allows the use of weights for examples, where the weights alter the Information Gain measure by using sums of weights instead of counts of examples. So instead of counting the number of examples with attribute value *a* and class *k*, the weights assigned to these examples would be summed and used in equation (2.1).

C4.5's use of weights has been utilized to incorporate misclassification costs, by overriding the weight initialization method. For example if the cost to misclassify a faulty example from the example dataset in Table 1 is 5, those examples belonging to class 'faulty' could be allocated the weight of 5, and examples belonging to class 'not faulty' could have the weight of 1, so that more weight is given to those examples with the higher misclassification cost. *C4.5CS* is one such algorithm which utilizes this use of weights.

The method of computing initial weights by *C4.5CS* is similar to that of the *GINIAlteredPriors* algorithm developed by Breiman et al. (1984) and Pazzani et al. (1994). When presented with the same dataset, both methods would produce the same decision tree. However Ting (1998) observes that the method which alters the priors would perform poorly as pruning would be carried out in a cost insensitive way, whereas the *C4.5CS* algorithm uses the same weights in its pruning stage. In his experiments with a version which replicates Breiman et al. (1984)'s method, C4.5(π') performs worse that the *C4.5CS* algorithm. He explains this result as owing to different weights in the tree growing stage and the pruning stage.

The sum of all the weights for class *j* in the *C4.5CS* algorithm will be equal to N. The aim of *C4.5CS* is to reduce high cost errors by allocating the highest weights to the most costly errors so that C4.5 concentrates on reducing these errors.

C4.5CS (Ting 1998, 2002):

$$weight_j = cost(j)\frac{N}{\sum_i cost(i)N_i} \quad (3.10)$$

where cost(j) and cost(i) are as defined by equation (3.8).

MaxCost (Margineantu and Dietterich 2003): $weight_j = max_{1 \le i \le k} C_{ji}$ (3.11)

AvgCost (Margineantu and Dietterich 2003): $weight_j = \frac{\sum_{i=1, i \neq j}^{k} C_{ji}}{(k-1)}$ (3.12)

These latter two algorithms have been designed to solve multi-class problems so the cost matrices involved are not the usual 2 x 2 grids presented when solving two class problems. Instead a *k* x *k* matrix is used, the diagonal cells containing the cost of correctly classifying an example, usually zero although for some domains it could well be greater than zero.

Table 5 presents an example of a cost matrix of a dataset where *k* = 4. The diagonal cells have been assigned zero therefore a correct classification results in zero cost. Two algorithms developed by (Margineantu and Dietterich 2003) use this cost matrix directly to compute initial weights. *MaxCost* uses the worst case cost of misclassifying an example. The maximum value within a column is considered to be the worst case cost of misclassifying an example. For instance, the weight of all class 1 examples will be assigned 100 as that is the maximum misclassification cost in the column corresponding to class 1. *AvgCost* calculates the average cost of misclassifying an example for its weight. Each weight is computed as the mean of the off-diagonal cells in the corresponding column. Using this algorithm, class 1

examples are assigned 35.6. These two algorithms are considered more efficient than others of this type (Margineantu and Dietterich 2003).

Predicted class	Correct Class			
	1	**2**	**3**	**4**
1	0	10	2	5
2	100	0	5	2
3	5	2	0	50
4	2	5	25	0

Table 5 Example of a cost matrix of a four class problem

Margineantu and Dietterich (2003) also suggest an alternative way of setting the weights, called *EvalCount*, where an accuracy-based decision tree is first induced and then used to obtain the weights. The training data is sub divided into a sub training set and a validation set. The sub training set is then used to grow an accuracy based decision tree. Using this decision tree, the cost of misclassification for each class on the validation set is then measured using the cost matrix. The weight allocated to a training example is then set to the total cost of misclassifying an example of that class.

3.1.1.2 Direct use of costs. Instead of adapting the information gain to include costs, a number of algorithms utilize the cost of misclassification directly as the selection criteria. These algorithms can be subdivided into two groups: those that only use misclassification costs and those which also include test costs.

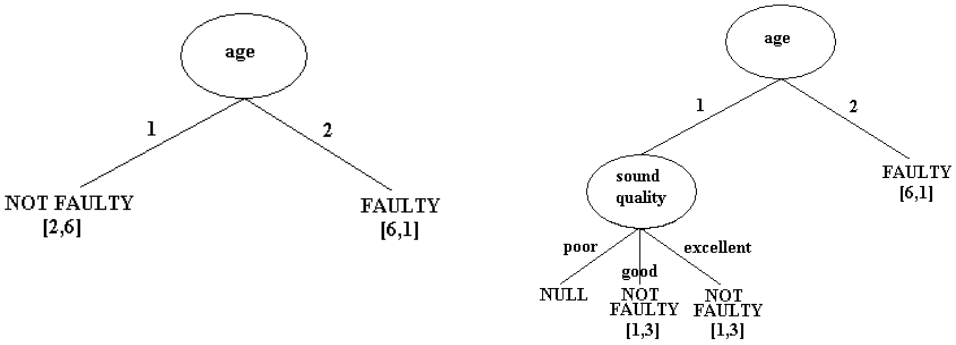
The central idea with these algorithms is to calculate the expected cost if an attribute is used to divide the examples, compared with the expected cost if there is no further division (i.e. a leaf is assumed). The attribute that results in the most reduction is then selected to divide the

examples. Of course, if none of the attributes results in a reduction, then a leaf node is created.

Cost-Minimization (Pazzani et al. 1994), *Decision Trees with Minimal Cost* (Ling et al. 2004) and two adaptations *Decision Trees with Minimal Cost under Resources Constrain* (Qin et al. 2004) and *CSTree* (Ling et al. 2006a) use either misclassification costs or a combination of misclassification costs and test costs to partition the data. *Cost-Minimization*, the simplest of these chooses the attribute which results in the lowest misclassification costs.

One of the main algorithms to use costs directly in order to find the attribute on which to partition data, is *Decision Trees with Minimal Cost* developed by Ling et al. (2004), spawning other adaptations. Expected cost is calculated using both misclassification costs and test costs aiming to minimize the total cost. An attribute with zero or smallest test cost is most likely to be the root of the tree, thus attempting to reduce the total cost. This algorithm has been developed firstly to minimize costs and secondly to deal with missing values in both the training and testing data. In training, examples with missing values remain at the node representing the attribute with missing values. In a study comparing techniques by Zhang et al. (2005), it was concluded that this was the best way to deal with missing values in training examples. How and whether to obtain values during testing are solved by constructing testing strategies and are discussed additionally in Ling et al. (2006b).

To illustrate what happens when only the costs (i.e., no information gain) are used to select attributes, consider the application of the *DT with MC* algorithm to the examples in Table 1, where in addition to the test costs we assume the misclassification costs of 50 and 200 for the faulty and not faulty class respectively.



5(a) Tree from DT with MC 5(b) Tree if left branch is expanded.

Figure 5 Decision tree when *DT with MC* has been applied to dataset in Table 1

Figure 5(a) shows the tree induced by *DT with MC* algorithm, which is very different from the cost-sensitive tree produced by *EG2* (Figure 4) and from the tree produced by ID3 (Figure 1). This algorithm employs pre-pruning, that is, it stops splitting as soon as there is no improvement. Figure 5(b) shows a partial tree obtained, if the left branch was expanded further. The additional attribute that would lead to the least cost is sound quality, with a total cost of 220 units since there are still two faulty examples misclassified but there is the extra cost of 120 units for testing Sound Quality (i.e., 8 examples each costing 15 units). However, the cost without splitting is 100 units (i.e., 2 faulty examples misclassified, with misclassification cost of 50) and hence, in this case, the extra test is not worthwhile.

Ling et al. (2006b) use the algorithm developed in Ling et al. (2004) in a lazy learning framework in order to use different test strategies to obtain missing values on test data and to address problems of delayed tests. Using expected total cost, a tree is induced for each test example using altered test costs, whereby test costs are reduced to zero for examples with known values, thus making them a more desirable choice.

Ling et al. (2004)'s algorithm is further adapted into *CSTree* which does not take into account test costs, using only misclassification costs (Ling et al. 2006a). *CSTree* deals with two-class problems and estimates the probability of the positive class using the relative cost of both classes and uses this to calculate expected cost.

A different and perhaps more extensive idea is by Qin et al. (2004), who develop an adaptation of the Ling et al. (2004) algorithm *Decision Trees with Minimal Cost under Resource Constrains*. Its purpose is to trade off between target costs (test costs and misclassification costs) and resources. Qin et al. (2004) argue that it is hard to minimize two performance metrics and it is not realistic to minimize both of them at the same time. So they aim to minimize one kind of cost and control the other in a given budget. Each attribute has two costs, test cost and constrain, likewise each type of misclassification has a cost and a constrain value. Both these values are used in the splitting criteria, to produce a target-resource cost decision tree (Qin et al. 2004) and used in tasks involving target cost minimization (test cost) and resources consumption for obtaining missing data.

Decision Tree with Minimal Costs under Resource Constrain:

$$\mathrm{ICF_A} = (\mathrm{T} - \mathrm{TA})/Constrain_A \tag{3.13}$$

$$Constrain_A = (N - o) * r_A + p * C_{ij}(r) + n * C_{ji}(r) + o * C_{ji}(r) \tag{3.14}$$

here T is the misclassification cost before splitting, TA is the expected cost if attribute A is chosen, r_A, Cij(r) and Cji(r) are the resource costs for false negatives and false positives respectively, p is the number of positive examples and n the number of negative examples and o the number of examples with missing attribute value.

A different approach than simply using the decision tree produced using direct costs, is suggested by Sheng and Ling (2005), a hybrid cost-sensitive decision tree. They develop a hybrid between decision trees and Naïve Bayes, *DTNB* (Decision Tree with Naïve Bayes). Decision trees have a structure which is used to collect the best tests but ignores, when

classifying, originally known attribute values not appearing in the path taken by a test example. It is argued by Sheng and Ling (2005) that any value is available at a cost, if values are available at the testing stage, these might be useful in order to reduce misclassification costs and to ignore them would be wasting available information. Naïve Bayes can use all known attribute values for classification but has no structure to determine which tests to perform and in what order should they be carried out in order to obtain unknown attribute values. The *DTNB* algorithm aims to combine the advantages of both techniques.

A decision tree is built using expected cost reduction using the sum of test costs and expected misclassification costs to determine whether to further split the data and on what attribute. Simultaneously a cost-sensitive Naïve Bayes model using Laplace correction and misclassification costs is hidden at all nodes including leaves and is used for classification only of the test examples. The decision tree supplies the sets of tests used in various test strategies and the Naïve Bayes model, built on all the training data, classifies the test examples, thus overcoming problems caused by segmentation of data, that is the reduction of data at lower leaves, and making use of all attributes with known values but which have not been selected during induction so that no information once obtained, is wasted. In experiments, this hybrid method proved to be better in combination than the individual techniques (Sheng and Ling 2005).

3.1.1.3 Linear and non-linear decision nodes. Most of the early algorithms handle numeric attributes by finding alternative thresholds, resulting in univariate or axis-parallel splits. A number of authors have suggested that this is not sufficiently expressive and adopted more sophisticated multivariate splits. These methods still adopt the top-down decision tree

induction process and the primary difference between them, which we summarize below, is whether they adopt linear or non-linear splits and how they obtain the splits.

The *LMDT* algorithm (Draper et al. 1994) was one of the first to go beyond axis-parallel splits. This algorithm aims to develop a decision tree whose nodes consist of Nilsson's (1965) linear machines. A linear machine aims to learn the weights of linear discriminants. Before looking at the *LMDT* algorithm, it is worth understanding the concept of a linear machine, which is central to the *LMDT* algorithm. The following figure summarizes the structure of a linear machine.

Each function $g_i(x)$ aims to represent a class i in a winner takes all fashion. A weight w_{ij} represents the coefficient of x_j for the i^{th} linear discriminant function. The training procedure involves presenting an example x that belongs to a class i. If the example is misclassified, say into class j, then the weights of the j^{th} machine can be decreased and the i^{th} machine increased, i.e.:

$$W^i = W^i + c.x$$
$$W^j = W^j - c.x \qquad (3.15)$$

where c is a correction factor, and the W^i and W^j are the weight vectors for the i^{th} and j^{th} linear discriminants.

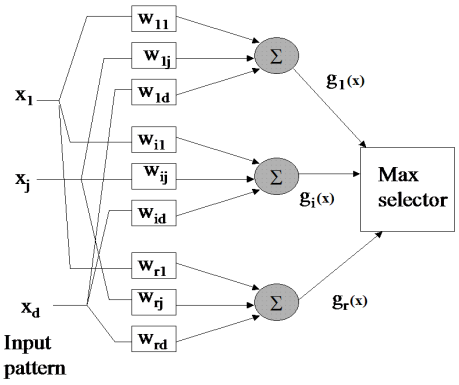


Figure 6 Linear Machine

When the classes are linearly separable, the use of a constant correction rate (i.e. as in a perceptron) is sufficient to determine a suitable discriminant and this simple procedure converges. However, in general, the classes may not be linearly separable and the above procedure may not converge. Draper et al. (1994) overcame this problem by utilizing a thermal training procedure developed by Frean (1990). This involved using an annealing parameter β to determine the correction factor *c* as follows:

$$c = \beta^2 / \beta + k \quad \text{where } k = (W_j - W_i)^T x / 2x^T x. \tag{3.16}$$

where W_j is the weight vector of the i^{th} discriminant function that represents the true class of the example, and W_j is the weight vector of the j^{th} discriminant function that represents the class in which the example is misclassified.

LMDT is altered to make it cost-sensitive by altering its weight learning procedure, with the aim of reducing total misclassification costs. In the modified version, it samples the examples based on the cost of misclassifications made by the current classifier. The training procedure is initialized for each class using a variable 'proportion$_i$', for each class *i*. Next, if the stopping criterion is not met, the thermal training rule trains the linear machine and if the examples have been misclassified, the misclassification cost is used to compute a new value for each 'proportion$_i$'.

An alternative approach to obtaining linear splits, taken in the *LDT* system (Vadera 2005b), is to take advantage of discriminant analysis which enables the identification of linear discriminants of the form (Morrison 1976; Afifi and Clark 1996):

$$(\mu_1 - \mu_2)\Sigma^{-1}x - \frac{1}{2}(\mu_1 - \mu_2)\Sigma^{-1}(\mu_1 + \mu_2) \leq \ln\left(\frac{C_{21}P(C_2)}{C_{12}P(C_1)}\right) \tag{3.17}$$

where $\boldsymbol{x}$ is a vector representing the new example to be classified, μ_1, μ_2 are the mean vectors for the two classes, Σ is the pooled covariance matrix, and $P(C_i)$ is the probability of an example being in class C_i.

Theoretically, it can be shown that equation (3.17) minimizes the misclassification cost when $\boldsymbol{x}$ has a multivariate normal distribution and when the covariance matrices for each of the two groups are equal.

This trend of moving towards more expressive divisions is continued in the *CSNL* system (Vadera 2010) that adopts non-linear decision nodes. The approach also utilizes discriminate analysis, and adopts following split that minimizes cost provided the class distributions are multivariate normal:

$$-\frac{1}{2}x^t(\sum_1^{-1}-\sum_2^{-1})x\ +(\mu_1^t\sum_1^{-1}-\mu_2^t\sum_2^{-1})x-k\geq \ln\left(\frac{C_{21}P(C_2)}{C_{12}P(C_1)}\right)$$

$$k=\ \frac{1}{2}ln\ (\frac{|\sum_1|}{|\sum_2|})+\frac{1}{2}(\mu_1{}^t\sum_1^{-1}\mu_1-\mu_2{}^t\sum_2^{-1}\mu_2) \qquad (3.18)$$

where x is a vector representing the example to be classified, μ_1, μ_2 are the mean vectors for the two classes, $\sum_1$, $\sum_2$ are the covariance matrices for the classes and $\sum_1^{-1}$, $\sum_2^{-1}$ the inverses of the covariance matrices.

Given that the multivariate assumption may not hold in practice, it may be that utilization of a subset of variables could lead to more cost-effective splits, and hence several strategies for subset selection are explored. One strategy, explored in Vadera (2005a), is to attempt all possible combinations and select the subset that minimizes cost. However, this strategy is not particularly scalable and results in trees that are difficult to visualize. An alternative strategy, explored in (Vadera 2010), selects two of the most informative features, as measured by information gain, and uses the above equation (3.18) to obtain non-linear divisions.

3.1.2 Post construction

If costs are unknown at training time they cannot be used for inducing a tree. Additionally if costs are likely to change, this would mean inducing a tree for every different combination of

costs. Hence, various authors have explored how misclassification costs can be applied after a tree has been constructed.

One of the simplest of ways is to change how the label of the leaf of the decision tree is determined. *I-gain Cost-Laplace Probability* (Pazzani et al. 1994) uses a Laplace estimate of the probability of a class given a leaf shown in equation (3.19). If there are N_i examples of class *i* at a leaf and *k* classes then the *Laplace* probability of an example being of class *i* is:

$$P(i) = \frac{N_i + 1}{k + \sum_{y=1}^{k} N_y} \tag{3.19}$$

When considering accuracy only, an example is assigned to the class with the lowest expected error. To incorporate costs, the class which minimizes the expected cost of misclassifying an example into class *j* is selected, where the expected cost is defined by:

$$Expected\ Cost\ of\ Misclassification\ into\ class\ j = \sum_i C_{ij}\, P(i) \tag{3.20}$$

Ferri et al. (2002), propose a post construction method based on Receiver Operating Characteristics (ROC) (Swets et al. 2000). ROC facilitates comparison of alternative classifiers by plotting their true positive rate (on the y axis) against their false positive rate (on the x axis). Figure 7 shows an example ROC, where the true and false rates of four classifiers are plotted. The closer a classifier is to the top left hand corner, the more accurate it is (since the true positive rate is higher and the false positive rate smaller).

The convex hull created from the points (0,0), the four classifiers and (1,1) represents an optimal front. That is, for any classifier below this convex hull, there is a classifier on the front that is less costly.

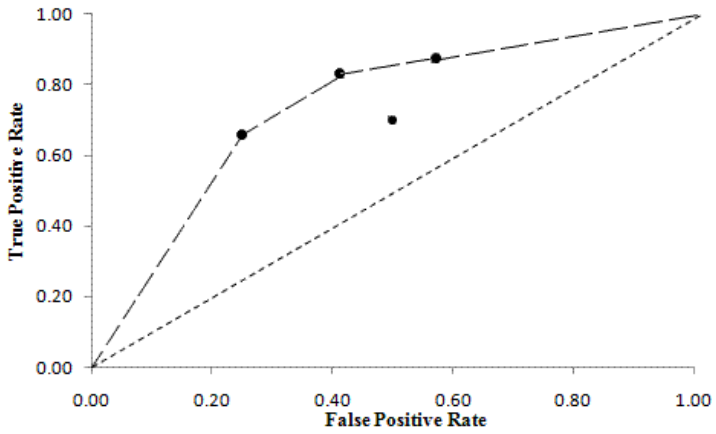


Figure 7 Example ROC

The idea behind Ferri et al. (2002)'s approach is to generate the alternative classifiers by considering all possible labellings for the leaf nodes of a tree. For a tree with *m* leaf nodes, and a two class problem, there are 2^m alternative labels, which could be computationally expensive. However, Ferri et al. (2002) shows that for a two class problem, if the leaves are ordered by the accuracy of one of the classes, then only $m+1$ alternative labellings are needed to define the convex hull, where the j^{th} node of the i^{th} labelling, $L_{i,j}$, is defined by:

$$L_{i,j} = \begin{cases} -ve \ if \ j < i \\ +ve \ if \ j \geq i \end{cases} \tag{3.21}$$

The convex hull formed by these labellings can then be used to determine the most optimal classifier once the costs of misclassification are known.

3.2 Multiple tree, non-greedy methods for cost-sensitive decision tree induction

Greedy algorithms have the potential to suffer from local optima, and hence an alternative direction of research has been to develop algorithms that generate and utilize alternative trees. There are three common strands of work: Section 3.2.1 describes the use of genetic algorithms, Section 3.2.2 describes methods for boosting and bagging, and Section 3.2.3 describes the use of stochastic sampling for developing anytime and anycost frameworks.

3.2.1 Use of Genetic Evolution for Cost-Sensitive Tree Induction

Several authors have proposed the use of genetic algorithms to evolve cost-effective decision trees (Turney 1995). Just as evolution in nature uses survival of the fittest in order to produce next generations, a pool of decision trees are evaluated using a fitness function, the fittest retained and combined to produce the next generation repeatedly until a cost-effective tree is obtained. This section describes the algorithms that utilize evolution, which vary in the way they represent, generate, and measure the fitness of the trees.

One of the first systems to utilize GAs was Turney's (1995) *ICET* system *(Inexpensive Classification with Expensive Tests. ICET* uses C4.5 but with *EG2's* cost function to produce decision trees, in Section 3.1.1.1.

Its populations consists of individuals with the parameters C_{Ai}, ω, and CF, where C_{Ai}, ω are biases utilized in equation (3.1) and CF is a parameter used by C4.5 for determining the aggressiveness of pruning.

ICET begins by dividing the training set of examples into two random but equal parts: a sub-training set and a sub-testing set. An initial population is created consisting of individuals with random values of C_{Ai}, ω, and CF. C4.5, with the *EG2*'s cost function, is then used to generate a decision tree for each individual. These decision trees are then passed to a fitness function to determine fitness. This is measured by calculating the average cost of classification on the sub-testing set.

The next generation is then obtained by using the roulette wheel selection scheme, which selects individuals with a probability proportional to their fitness. Mutation and crossover are used on the new generation and passed through the whole procedure again. After a fixed number of generations (cycles) the best decision tree is selected. *ICET* uses the GENEtic Search Implementation System (GENESIS, Grefenstette (1990)) with its default parameters including a population size of 50 individuals, 1000 trials and 20 generations.

More recently, Kretowski and Grześ (2007) describe *GDT-MC (Genetic Decision Tree with Misclassification Costs)*, an evolutionary algorithm in which the initial population consists of decision trees that are generated using the usual top down procedure, except that the nodes are obtained using a dipolar algorithm. That is, to determine the test for a node, first two possible examples from the current dataset are randomly chosen such that they belong to different classes. A test is then created by randomly selecting an attribute that distinguishes the two examples. Once a tree is constructed, it is pruned using a fitness function. The fitness function used in *GDT-MC* aims to take account of the expected misclassification cost as well as the size of trees and takes the form (Kretowski and Grześ, 2007):

$$Fitness\ of\ tree = \left(1 - \frac{EC}{MC}\right)(1 + \gamma . TS) \quad (3.22)$$

where EC is the misclassification cost per example, MC is the maximal possible cost per example, TS is the number of nodes in the tree and γ is a user provided parameter that determines the extent to which the genetic algorithm should minimize the size of the tree to aid generalization.

The genetic operators are similar in principle to the cross-over and mutation operators, except that they operate on trees. Three cross-over like operators are utilized on two randomly selected nodes from two trees:

- exchange the sub-trees at the two selected nodes.
- if the types of tests allow, then exchange just the tests.
- exchange all sub-trees of the selected nodes, randomly selecting the ones to be exchanged.

The mutation operators adopted allow a number of possible modifications of nodes, including replacing a test with an alternative dipolar test, swapping of a test with a descendent node's test, replacement of a non-leaf node by a leaf node, and development of leaf node into a sub-tree. A linear ranking scheme, coupled with an elitist selection strategy, is utilized to obtain the next generation (Michalewicz 1996).[3]

The *ECCO (Evolutionary Classifier with Cost Optimisation)* system (Omelian 2005) adopts a more direct use of genetic algorithms by mapping decision trees to binary strings and then adopting the standard cross-over and mutation operators over binary strings. Attributes are represented by a fixed size binary string, so for example 8 attributes are coded with 3 bits. Numeric attributes are handled by seeking an axis parallel threshold value that maximizes

[3] The elitist strategy ensures that a few of the fittest are copied to the new generation, and the linear ranking strategy ensures some diversity and avoids the fittest don't dominating the evolution to early in the evolution.

information gain, thereby resulting in a binary split. The mapping between a tree and its binary string is achieved by assuming a fixed size maximal tree where each node is capable of hosting an attribute which has the most features.[4] Figure 8 illustrates the mapping for a problem where the attributes have two features only. Such a maximal tree is then interpreted by mapping the nodes to attributes, assuming that the branches are ordered in terms of the features. In addition, mutation may result in some nodes with non-existent attributes, which are also translated to decision nodes.

A tree is then populated with the examples in a training set and each leaf node labelled with a class that minimizes the cost of misclassification. A version of the minimum error pruning algorithm that minimizes cost instead of error is used for pruning. The fitness measure used is the expected cost of classification, taking account of both the cost of misclassification and the cost of the tests. Once genes are mapped to decision trees and pruned, and their fitness obtained, the standard mutation and cross-over operators applied, a new generation of the fittest is evolved and the process repeated a fixed number of cycles. Like *ICET*, *ECCO* adopts the GENESES GA system and adopts its default parameters.

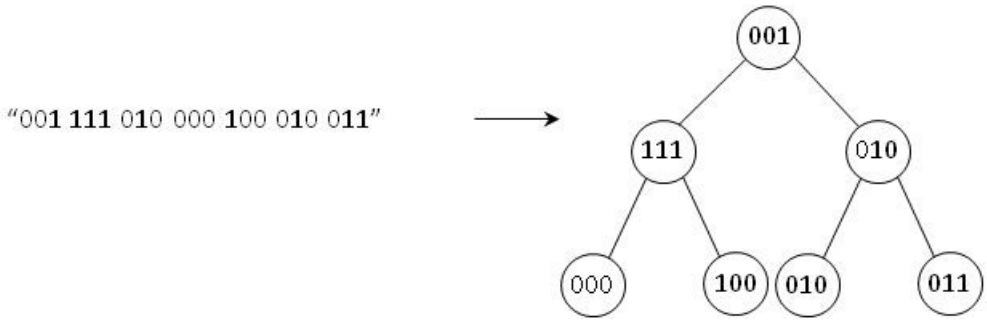


Figure 8 Illustration of mapping

[4] The approach works in general for an attribute with more than two features

Li et al. (2005) take advantage of the capabilities of Genetic Programming (GP), which enable representation of trees as programs instead of bit strings, to develop a cost-sensitive decision tree induction algorithm. They use the following representation of binary decision trees as programs, defined using BNF (Li et al. 2005):

<Tree> :: "if-then-else" <Cond><Tree><Tree> | Class

<Cond> :: <Cond> "And" <Cond> | <Cond> "Or" <Cond>

| Not <Cond> | Variable<RelationOperation>Threshold

RelationOperation ::= ">" | "<" | "="

Unlike *GDT-MC,* which utilizes specialized mutation and crossover operators, Li et al. (2005) adopt the standard mutation and crossover operators of genetic programming. A tournament selection scheme, in which four individuals are selected randomly with a probability proportional to their fitness, compete to move to the next generation. The fittest of the four is copied to the pool for the next generation and this tournament process repeated to produce the complete mating pool for the next generation. The fitness function employed is also different from *ICET*, *ECCO* and *GDT-MC*. Unlike, these methods, which utilize expected cost, Li et al (2005) propose the following fitness function that is based on the principle that a cost-effective classifier will maximize accuracy (RC) but minimize the false positive rate (RFP):

$$\textit{Constraint Fitness Function} = W_{rc'} * RC - W_{rfp} * RFP \qquad (3.23)$$

Experimentation with this function leads them to the following additional constraint to ensure that accuracy of one of classes is not compromised when the costs of misclassifications are significantly imbalanced:

$$W_{rc} = 1 \text{ if } C_+ \in (P_{min}, P_{max}), 0 \text{ otherwise,} \quad (3.24)$$

where C_+ is the proportion of examples predicted to be positive, and the P_{min} and P_{max} define the expected range for C_+ that is provided by a user.

3.2.2 Wrapper methods for cost-sensitive tree induction

A significant amount of research has been done on accuracy based classifiers, and instead of developing new cost-sensitive classifiers or adapting them as described above, an alternative strategy is to develop wrappers over accuracy based algorithms.

This section describes two approaches for utilizing existing accuracy based algorithms. Section 3.2.2.1 describes methods based on boosting, where an accuracy based learner is used to generate an improving sequence of hypotheses and Section 3.2.2.2 describes methods based on bagging that are based on generating and combining independent hypotheses. Section 3.2.2.3 describes a method which implicitly includes alternative hypotheses but in one structure.

3.2.2.1 Cost-Sensitive Boosting. Boosting involves creating a number of hypotheses h_t and then combining them to form a more accurate composite hypothesis of the form (Schapire 1999; Meir and Rätsch 2003):

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \quad (3.25)$$

where α_t indicates the extent of weight that should be given to $h_t(x)$.

One of the first practical boosting methods, AdaBoost (Adaptive Boosting) works by generating $h_i(x)$ in sequential trials by using a learner on weighted examples that reflect their importance (Freund and Schapire 1996). It begins by assigning weights of *1/N* to each example. At the end of each sequential trial, these weights are adjusted so that the weights of misclassified examples are increased, but the weights of correct examples decreased. After a fixed number of cycles, a sequence of trees or hypotheses h_i is available and can be combined to perform classification. The final classification is based on selecting the class that results in the maximum weighted vote as defined by equation (3.25). There are different versions of AdaBoost with specific weight update rules (e.g., Freund and Schapire (1997), Bauer and Kohavi (1999), Schapire and Singer 1998, 1999)). For example, one version that is based on a weak learner capable of producing hypotheses h_t that return a confidence rating in the range (-1,1) uses the following update rule (Schapire 1999):

$$\alpha_t = \tfrac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{3.26}$$

$$w_{t+1}(x) = \frac{w_t(x)\exp\left(-\alpha_t\, y h_t(x)\right)}{Z_t}$$

where the Z_t is used to normalize the weights so they add up to 1.

Thus, AdaBoost consists of three key steps: the initialization, the weight update equations, and the final weighted combination of the hypotheses. The literature contains a number of algorithms that adapt these three steps of AdaBoost to develop cost-sensitive boosting algorithms.

In particular, Ting and Zheng (1998a), which was one of the first studies to utilize boosting for cost-sensitive induction, proposed two adaptations: an algorithm called *UBoost* (*Boosting*

with Unequal Instance Weights) and another called *Cost-UBoost* (*UBoost with Cost-Sensitive adaptation*).

UBoost utilizes AdaBoost, except that the weights for each example x, of class *j* are initialized to the cost of misclassifying an example of class *j*, and normalized[5] :

$$w_0(x) = cost(j) \tag{3.27}$$

The cost of misclassifying an example of class *i*, denoted by *cost(i)* is defined by Ting and Zheng (1998a) as in equation (3.8). Below, we also use the notation *cost(x)* to denote the cost of misclassifying an example *x*.

In addition, the composite classification rule of equation (3.25) is adapted to first work out the expected cost of classifying an example $EC_j(x)$, into class *j* using the combined hypotheses:

$$EC_j(x) = \sum_{t=1}^{T} \alpha_t EC(x, j, h_t) \tag{3.28}$$

where $EC(x,j,h_t)$ is the expected cost if the example *x* is classified in class *j* based on the distribution of examples in the leaf node of the tree h_t that leads to the classification $h_t(x)$.

UBoost then selects the class *j* that results in the minimum expected cost $EC_j(x)$.

Ting and Zheng (1998a) also propose a method *Cost-UBoost* that extends *UBoost* by also amending the weight update procedure to take account of costs, so that:[3]

[5] The presentation here assumes that the normalisation of the weights by a factor is Z_t is done at the end of a trial, therefore simplifying the equations.

$$w_{t+1}(x) = w_t(x).\beta(y', y) \tag{3.29}$$

where *y* is the actual class and y' is the predicted class for an example *x* and β is defined by:

$$\beta(y', y) = \begin{cases} C_{yy'} \, when \, y' \neq y \\ 1 \quad when \, y' = y \end{cases} \tag{3.30}$$

The empirical trials conducted by Ting and Zheng (1998a) suggest that *Cost-UBoost* performs better than *UBoost* in terms of minimizing costs of misclassification for two class problems. However, they note that this advantage reduces for multi-class problems and suggest that this is owing to the mapping of different costs of misclassification into a single misclassification cost by equation (3.8). Later in this section, we describe the more recent work of Abe et al. (2004), and Lozano and Abe (2008) that develops theoretical foundations for multi-class cost-sensitive boosting problems.

In a follow up study, Ting (2000b) propose further variations, named *CSB0*, *CSB1*, *CSB2* and compare their performance to another variation of AdaBoost, known as *AdaCost* (Fan et al. 1999). *CSB0* is essentially the *Cost-UBoost* algorithm described above, while *CSB1*, *CSB2* and *AdaCost* utilize increasingly sophisticated weight update functions for weak learners that produce the confidence in its prediction $h_t(x) \in (0, 1)$ (Ting 2000b):

$$CSB1: \quad w_{t+1}(x) = w_t(x)\beta(y', y)\exp(-\delta h_t(x)) \tag{3.31}$$

$$CSB2: \quad w_{t+1}(x) = w_t(x)\beta(y', y)\exp(-\delta h_t(x)\alpha_t) \tag{3.32}$$

$$AdaCost: \quad w_{t+1}(x) = w_t(x)\exp(-\delta h_t(x)\alpha_t\beta'(y, y')) \tag{3.33}$$

where δ is -1 if the example is misclassified and +1 if classified correctly, and a α_t is defined as derived in (Shapire and Singer 1999):

$$\alpha_t = \tfrac{1}{2} ln\left(\frac{1+r_t}{1-r_t}\right) \tag{3.34}$$

and with r_t defined as follows for the *CSB* family:

$$r_t = \frac{1}{N}\sum_{x\epsilon X} \delta w_t(x) h_t(x) \tag{3.35}$$

As well as the update equation, the r_t and cost adjustment function β' are defined differently for *AdaCost*:

$$r_t = \sum_{x\epsilon X} \delta w_t(x) h_t(x) \beta'(y, h(x)) \tag{3.36}$$

$$\beta'(y', y) = \begin{cases} 0.5\, cost(x) + 0.5, & when\ y' \neq y \\ -0.5\, cost(x) + 0.5, & when\ y' = y \end{cases} \tag{3.37}$$

Ting (2000b) evaluates these methods empirically and concludes that the introduction of the α_t in *CSB2* does not lead to a significant improvement and the additional parameters used in *AdaCost* are not particularly effective either. *CSB1* produces more cost-effective results than *AdaCost* in 30 runs while *AdaCost* performs better in 11 runs. Surprisingly, the evaluations also suggest that AdaBoost produces better results than its cost-sensitive version *AdaCost*, which Ting (2000b) attributes to the particular definition of β' that allocates a relatively low reward (penalty) when high cost examples are correctly (incorrectly) classified. This is in contrast to the results presented in (Fan et al. 1999), where *AdaCost* produces better results than AdaBoost when the Ripper learner is used instead of C4.5 as the base learner.

The above adaptations of boosting presume that costs are well-defined in advance. Merler et al. (2003) argue that in medical applications, the costs of false positives or false negatives can only be approximate, and further that during the classification process there two separate phases. In the first phase, the aim is to ensure that the classifier is sensitive and the true

positives are maximized whilst the specificity of a classifier is retained within acceptable bounds. In a second phase, a specialist medical consultant would examine the identified positives more carefully, filtering out the false negatives. Hence, for this type of application, they develop a boosting algorithm, *SSTBoost* (*Sensitivity-Specificity tuning Boosting*) that adapts AdaBoost so that the error for the i^{th} example is defined in terms of measures of sensitivity and specificity:

$$\varepsilon_i = (1 - Sensitivity)\pi_{+1}c_{+1} + (1 - Specificity)\pi_{-1}c_{-1} \quad (3.38)$$

where π_{+1} π_{-1} are the class priors and c_{+1}, c_{-1} are the costs of misclassification of the two classes. Sensitivity is the true positive rate and specificity is true negative rate.

With this definition of error, they use equation (3.26) for α_t :

$$\alpha_t = \left(\frac{1}{2}\right) \ln \left(\frac{1-\varepsilon_i}{\varepsilon_i}\right) \quad (3.39)$$

The weight update equation takes the form:

$$w_{t+1}(x) = \begin{cases} \mathrm{w_t(x)exp}\,(-\alpha_t(2 - cost(x))), & if example\ x\ is\ classified\ correctly \\ w_t(x)\exp\left(\alpha_t cost(x)\right), & if\ example\ x\ is\ classified\ incorrectly \end{cases} \quad (3.40)$$

Given specific costs for misclassification, this adaptation of AdaBoost, results in a classifier with a particular sensitivity and specificity. To enable a search for a classifier in a target region of sensitivity and specificity, they relate the costs of misclassifying a positive example, c_+, and cost of misclassifying negative example, c_-, in terms of a single parameter ω:

$$c_{+1} = \omega$$
$$c_{-1} = 2 - \omega \quad (3.41)$$

This then enables a search over ω by using the method of bisection to find a classifier that aims to be within a user specified region of sensitivity and specificity, meeting their application goals.

The above adaptations of AdaBoost amend the procedure to take account of costs. In contrast, as part of a study that aims to utilize boosting for estimating conditional class probabilities, Mease et al. (2007) describe how AdaBoost can be used directly to develop a procedure called *JOUS-Boost* to perform cost-sensitive boosting. They use a result owed to Elkan (2001) that shows that it is possible to change the distribution of the data to reflect the ratio of costs and such that applying boosting on this changed distribution results in minimization of cost. More specifically, given the cost of misclassification and number of examples of class 1 and class 2 are N_1, N_2 respectively, the distribution of the data is changed so that the number of examples N_1', N_2' of class 1 and 2, satisfy:

$$\frac{N'_1}{N'_2} = \frac{N_1 C_{12}}{N_2 C_{21}} \quad (3.42)$$

This change of distribution can be achieved by sampling the original data in a way that results in a smaller dataset (under-sampling) or a larger dataset (over-sampling). The sampling itself can be done with replacement, where a selected example is returned, or without replacement. Under-sampling can result in loss of data and over-sampling leads to duplication of examples. Mease et al. (2007) carry out experiments on both artificial and real data showing that the duplication owing to over-sampling leads to over-fitting when boosting. They then propose a variation, called *JOUS-Boost* (*Over/Under Sampling and Jittering*), that amends the

sampling process by adding noise to the features of any duplicated data and provide empirical evidence to show that this helps to reduce over-fitting when AdaBoost is used.

Most of the above algorithms are based on using boosting on two class problems. In two class problems, algorithms such as *UBoost* are able to set the weight of an example in proportion to the cost of misclassifying an example. However, for multi-class problems, an example could be misclassified into several classes, so determining the weight is less obvious. Several authors have proposed methods such as utilizing the sum or average of misclassification into the other classes (e.g. Breiman, et al (1984); Margineantu (2001)); though as noted above, Ting and Zheng (1998a) suggest that use of these methods may explain a reduction in the advantage gained by *Cost-UBoost* over *UBoost* in their empirical evaluations. Abe et al. (2004) also argue that these methods do not have a theoretical basis. Hence, they propose an alternative way of utilizing boosting, called *Gradient Boosting with Stochastic Ensembles* (*GBSE*), for multi-class problems. *GBSE* is motivated by first defining a stochastic hypothesis $H(y|x)$ for a class y for an example x based on the individual hypotheses $h_t(x)$ generated by (Abe et al. 2004):

$$H(y|x) = \frac{1}{T}\sum_{t=1}^{T} I(h_t(x) = y) \qquad (3.43)$$

If $H_t(y|x)$ is the composite hypothesis after round t of boosting, then it is formed by combining the previous composite hypothesis H_{t-1} with the new hypothesis, h_t, obtained in round t, weighted by α_t:

$$H_t(y|x) = (1 - \alpha_t t)H_{t-1}(y|x) + \alpha_t I(h_t(x) = y) \qquad (3.44)$$

where I(E) returns 1 if the expression E is true and 0 if E is false, α = 1/t, and initially H0(x|y) = 1/k, for a k-class problem.

This enables the definition of the expected cost of misclassification over the examples, and using gradient descent, Abe et al. (2004) then derive the following weight update rule, where $w_{x,y}$ is the weight associated with classifying example x in class y:

$$w_{x,y} = cost_{Ht-1}(x) - cost(x,y) \tag{3.45}$$

where $cost_{Ht-1}(x)$ is the expected cost of classifying example x by the composite hypothesis Ht-1(x).

However, existing boosting methods assume a single weight of importance per example, and not multiple weights, $w_{x,y}$. Hence, to utilize existing boosting methods, there needs to be a mapping to and from multiple weights to single weights per example. Abe et al. (2004) show that this mapping can be achieved by expanding an example $(x, y, (c_1, c_2, \ldots c_k))$, that has features x, class y, and costs of classification into class i defined by c_i, into k examples :

$$S = \{(x, y, max_j c_j - c_i) | i \in (1..k)\} \tag{3.46}$$

Abe et al. (2004) prove that minimizing cost over this expanded dataset is equivalent to minimizing the cost over the original multi-class data. They note that equation (3.45) can lead to negative weights $w_{x,y}$ which makes it difficult to utilize existing relational weak learners. They therefore transform the examples of equation (3.46) to the following form:

$$\left\{\left((x,y), I(w_{x,y} \geq 0), |w_{x,y}|\right) \middle| x \in X, y \in Y\right\} \tag{3.47}$$

A weak learner can be applied to these data and used to induce a relational hypothesis $h_t(x,y)$ and the composite hypothesis revised:

$$H_t(y|x) = (1 - \alpha_t t)H_{t-1}(y|x) + \alpha_t f_t(y|x) \quad (3.48)$$

where $f_t(y|x)$ converts the relational hypothesis to a stochastic form:

$$f_t(y|x) = \begin{cases} H_{t-1}(y|x), & if\ no\ examples\ x\ such\ that\ h(x,y) = 1 \\ \dfrac{I(h(x,y) = 1)}{|\{y \in Y | h(x,y) = 1\}|} & otherwise \end{cases} \quad (3.49)$$

This formulation defines the mapping needed for *GBSE* to use single weight boosting methods for multi-class problems.

A desirable property of any boosting algorithm is that it should converge and lead to the optimization of its objective. Although this has not been shown for *GBSE*, Abe et al. (2004) show that a variant, called *GBSE-T*, with a fixed α, and the following amendment of the *GBSE* weight update equation (3.45) converges exponentially:

$$w_{x,y} = \frac{cost_{Ht-1}(x)}{k} - cost(x,y) \quad (3.50)$$

In a follow up study, Lozano and Abe (2008) develop stronger theoretical foundations for cost-sensitive boosting in which they derive update equations for a family of methods, called *Cost-Sensitive Boosting with p-norm Loss* (*L_p-CSB*) for which they prove convergence. Like the above study, they use stochastic gradient boosting as the methodology, however, instead of aiming to minimize the expected cost of misclassification at each boosting round, they aim to minimize its approximation using the p-norm (Lozano and Abe 2008):

$$\frac{1}{|S|}\sum_{(x,y,w)\in S}(h(y|x)^p)cost(x,y) \quad for\ p\ \geq 1 \tag{3.51}$$

where S takes the expanded form defined as equation (3.46).

They use methodology similar to the derivation of *GBSE* and use gradient descent to show that optimizing this p-norm based objective leads to finding a hypothesis h_t at each round that minimizes (Lozano and Abe 2008, p507):

$$\sum_{x\in X}\sum_{y\in Y} w_{x,y}(I(h_t(x)=y)) \tag{3.52}$$

where

$$w_{x,y} = H_{t-1}(y|x)^{p-1}cost(x,y) \tag{3.53}$$

To facilitate the use of a relational weak learner, the examples are translated to the following form, into a similar form to equation (3.47) in *GSBE* (p508):

$$S = \{((x,y),l,w'_{x,y})|\ x\ \in X, y\in Y\ \} \tag{3.54}$$

However, the weights are different from *GSBE* and defined by:

$$w'_{x,y} = \begin{cases} \frac{w_{x,y}}{2}\ and\ l=0, for\ (x,y) when\ y\ does\ not\ correspond\ to\ the\ minimum\ cost\ class \\ \frac{\sum_{y\in Y'} w_{x,y}}{2}\ and\ l=1, where\ Y' is\ the\ set\ of\ all\ classes\ except\ the\ one\ with\ minimal\ cost \end{cases} \tag{3.55}$$

Once a weak learner is applied and a new hypotheses $h_t(x,y)$ obtained, the revised composite hypothesis is defined as in equation (3.48) with $f_t(x|y) = h_t(x,y)$. Lozano and Abe (2008)

prove that this scheme and its related family of schemes are guaranteed to minimize the p-norm based objective, providing a significant theoretical result and understanding of cost-sensitive boosting methods.

3.2.2.2 Cost-Sensitive Bagging. The main principle of bagging is that producing *n* re-samples of the dataset (with replacement), applying a learning procedure to each resample and aggregating the answers leads to better classifiers, particularly for learners that are not stable (Breiman 1996). This principle is used in *MetaCost* (Domingos 1999), which is one of the earliest systems to utilize cost-sensitive bagging. Thus, *MetaCost* re-samples the data several times and applies a base learner to each sample to generate alternative decision trees. The decisions made on each example by the alternative trees are combined to predict the class of each example that minimizes the cost and the examples relabelled. The relabelled examples are then processed by the base learner, resulting in a cost-sensitive decision tree.

Zadrozny et al. (2003a, 2003b) describe a method called *Costing* that, like *MetaCost* applies a base learner to samples of the data to generate alternative classifiers. However, the sampling process is significantly different from *MetaCost*. Each resample aims to change the distribution of the data so that minimizing error on the changed distribution is equivalent to minimizing cost on the original distribution (i.e., as described for *JOUS-Boost* above). Zadrozy et al. (2003a) argue that using sampling with replacement can lead to overtraining because of the potential for duplication, and sampling without replacement is also problematic since we can no longer assume that the examples selected are independent. Hence, to overcome these shortcomings, Zadrozny et al. (2003a) utilize rejection sampling (Von Neumann 1951) in which an example with cost *c* has a probability of c/Z of being accepted, where Z is chosen as the maximum cost of misclassifying an example. Once these

samples, which are proportional to the cost, are generated using rejection sampling, *Costing* applies a base learner to generate *m* classifiers whose outcomes can be aggregated to classify an example. Notice that, unlike *MetaCost*, there is no relabeling of the data in order to generate a single decision tree.

Lin and McLean (2000) develop an approach, in which they use different learners on the same training sample to generate alternative classifiers. As with *MetaCost*, they use the different classifiers to predict the class of each example. However, the labelling of an example *x*, by a classifier *j* is based on the risk of classification into two classes:

$$\text{Risk}_{1,j} = \quad \pi_2 * P(2|x,j) * C_{12} \qquad (3.56)$$

$$\text{Risk}_{2,j} = \quad \pi_1 * P(1|x,j) * C_{21}$$

The risk of classification of an example *x* into a class *c* is then defined as a weighted sum of the *m* classifiers:

$$\text{Risk}_{x,c} = \sum_{j=1}^{m} w_j R_{c,j} \qquad (3.57)$$

where w_j, which is the weight associated with classifier *j*, is the accuracy of the classifier on the training set. The class *c* that minimizes $\text{Risk}_{x,c}$ is used to label an example *x*.

Moret et al. (2006) describe a similar method to Lin and McLean (2000), called *Bagged Probability Estimation Trees (B-PETS*), but do not utilize the prior class probabilities, $\pi_{i,}$ in equation (3.56) and also estimate the P(i|x,j) using the distribution of examples in the leaf nodes and Laplace's correction (equation 3.19) which is known to produce better estimates.

Moret et al. (2006) also propose an alternative way of estimating P(c|x), the probability of an example *x* being in a particular class *c* that makes use of lazy option trees. A lazy option tree (LOT) is constructed for an example *x*, using the usual top-down process except there are two significant differences. First, since the example is known, only tests that are consistent with the example are considered at each node. Secondly, instead of selecting a single best test for each node, the first *k* best tests are also stored as alternative tests, leading to *k* sub-trees. An estimate of P(c|x) is then based on the *k* leaf nodes that the example *x* falls into. In addition, they also use re-sampling and bagging over lazy option trees (*B-LOT*s), to produce estimates of P(c|x). These estimates of P(c|x) can then be used to select the class that minimizes the risk based on the cost of misclassification.

Given that both *MetaCost* and AdaBoost each result in improved performance, it seems plausible that exploring a combination of the two methods could lead to further improvements. Ting (2000a) investigates this possibility by carrying out an empirical evaluation of two adaptations of *MetaCost*: one, called *MetaCost_A* where the base learner is AdaBoost, and a second, called *MetaCost_CSB*, where the base learner is *CSB0*. The results of the empirical evaluations suggest that there is little to be gained by embedding AdaBoost or *CSB0* within *MetaCos*t. Bagging is known to be particularly effective at reducing variance owing to an unstable base learner (Bauer and Kohavi 1998), which may provide an explanation of why using bagging over AdaBoost or CSB does not result in further improvements.

The comparison in Ting (2000a) also shows that using a cost-sensitive base learner for *MetaCost* does result in improvements over a using a cost-insensitive learner, which is also apparent in the empirical results presented in Vadera (2010).

3.2.2.3 Multiple Structures. Estruch et al. (2002) argue that generating alternative trees such as in boosting and bagging can consume significant space and therefore propose a structure, called *Multi-Tree*, which aims to implicitly include alternative trees. The central idea is to follow the usual top-down decision induction process, but instead of discarding alternative choices, these are stored as suspended nodes that could be expanded in the future (Ferri-Ramírez 2002; Rissanen 1978). Figure 9 shows a multi-tree for the example given Table 1 of 'Television Repair' dataset. The attributes that have been selected are presented in rectangles, and the suspended nodes, which are in circles, are linked by the dashed lines. The figure also includes the class distribution in each node and is given in the format (number of examples in 'faulty' class, number of examples in 'not faulty' class). A multi-tree can be expanded to include an additional tree by selecting a suspended node and developing it into a tree using the top-down process but retaining potential attributes as suspended nodes. Estruch et al. (2002) consider alternative methods of selecting which suspended node to expand and adopt a random selection scheme. Thus a multi-tree will implicitly include several trees each of which can be used for classification and whose outcomes can be combined to produce a weighted classification in the same manner to bagging.

Estruch et al. (2002) experiment with different ways of producing this weighted classification by taking advantage of the fact that different decision trees may share the same part of a multi-tree. A multi-tree is not as comprehensible as a single tree, and hence a method for extracting a single tree is developed.

In contrast to MetaCost, where a single tree is obtained by applying a base learner on the re-classified examples, a single tree is extracted by traversing a multi-tree bottom-up, and selecting those suspended nodes that agree the most with the outcomes of the multi-tree using

a randomly created dataset. They then utilize ROC, as described in Section 3.1.2, to take account of costs. Estruch et al. (2002) includes an empirical comparison which concludes that it is more efficient in comparison to bagging and boosting. The results for accuracy suggest that bagging produces better results at lower number of iterations while the use of multi-tree produces slightly better results beyond 200 iterations.

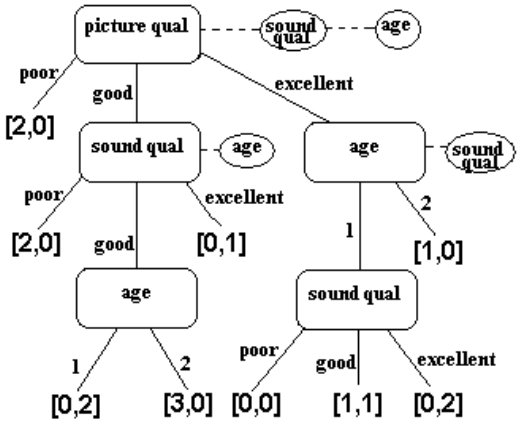


Figure 9 Multi-tree using the example dataset

3.2.3 Stochastic Approach

The greedy methods of induction of trees, described in Section 3.1, select an attribute after considering its immediate effect on the examples. Several authors have investigated the potential for utilizing a *k*-look-ahead strategy to select attributes by considering their effect deeper down a tree (e.g., Murthy and Salzberg (1995), Dong and Kothari (2001)). That is, for each attribute, a sub-tree of depth *k* is developed and the attribute that results in the best sub-tree is selected. Although increasing the look-ahead depth *k* has the potential for increasing the quality of a tree, as Esmeir and Markovitch (2004) point out, this also leads to an exponential increase in the time required for induction.

Hence, in their work they explore the use of stochastic sampling methods to assess the attributes and develop ACT (Esmeir and Markovitch 2007, 2008), a framework for anytime induction of cost-sensitive trees, and TATA (Esmeir and Markovitch 2010, 2011), an anycost framework for learning under limited budgets.

ACT (*Anytime Cost-sensitive trees*) (Esmeir and Markovitch 2007, 2008) uses a stochastic tree induction algorithm to generate *r* samples of sub-trees for each value of an attribute. The cost of each of these sub-trees is calculated using the training examples, and the minimum cost utilized as an estimate for the attribute value. The costs of the sub-trees for the attribute values are aggregated to estimate the cost of selecting an attribute, and the minimum cost attribute selected. The first of the *r* samples is generated using the *EG2* algorithm and the remaining samples are generated using a greedy top-down induction process except that the probability of selecting an attribute is proportional to its information cost measure as defined in *EG2* equation (3.1).

In experiments, *ACT* returned better results than *ICET* (Section 3.2.1) and *Decision Trees with Minimal Cost* (Section 3.1.1.2) (Esmeir and Markovitch 2008, p26).

In a more recent study, Esmeir and Markovitch (2011) note that minimizing the sum of test and misclassification costs implies that the costs should be on the same scale.[6] They argue that a more realistic goal would be to develop trees that minimise misclassification costs but subject to a constraint that the total cost of the tests utilised is no more than a specified cost.[7] The limit on test costs may be available prior to learning, after learning but before classification, or may be unavailable, leading to algorithms they term as pre-contract, contract

[6] As mentioned in section 4.1.1, Zhang et al. (2007), also make the same observation, though they adapt the measure used in the Performance algorithm to represent the trade-off between costs of tests and costs of misclassification.
[7] Greiner et al. (2002) provides some theoretical results for active learning under such budgets.

and interruptible classifiers. They develop a framework for algorithms for such situations, called *TATA (Tree classification AT Anycost)*, that is capable of reducing misclassification costs as the budget for using tests increases.

They develop this framework by first noting that existing top-down tree induction algorithms can be adapted so that the total test cost for any example will be no more than a pre-specified cost. This can be achieved during the tree induction process by only considering those attributes whose cost is below the current available budget, where the current budget is the initial budget less the cost of the attributes used from the root to the current node. Then, they adopt an approach similar to *ACT*, except that the *r* samples are obtained using an adapted version of C4.5 in which attributes that cost more than the available budget are excluded and attributes are selected stochastically with a probability proportional to their information gain. The samples for each available attribute are used to estimate the misclassification cost and the one with minimal misclassification cost selected. Given a maximum budget available for testing and a suitable sample size *r*, this achieves the requirements for a pre-contract algorithm.

For a contract algorithm, the budget for test costs in not available until the classification stage. To handle such applications, Esmeir and Markovitch (2011) propose inducing a sequence of trees, t_1, … t_k, which they term a repertoire, with respective budgets $c_1,\ldots,c_k$, where c_1 is set to the cost of the cheapest test, and c_k is set to the maximum cost, where all the tests are used. The number of trees, *k*, that are used, depends on the amount of time and memory available but also impacts on the time available for the number of stochastic samples, r, that are possible. The k trees could be obtained by discretizing the interval from c_1 to c_k into k-1 uniformly spread intervals or in a more sophisticated manner by repeatedly

using hill-climbing to subdivide an interval that has the largest gap in terms of expected errors and test cost budgets.[8]

To achieve the goals of an interruptible algorithm, where neither the budgets for learning or the total tests costs are available, they propose developing a repertoire of trees and then to start classification using the tree with the minimum possible budget, and then repeatedly moving on to the tree with the next higher budget until interrupted or reaching the final tree.

An empirical evaluation of *TATA* shows that misclassification costs reduce more rapidly with increasing budget when compared to *EG2*, an adapted version of *EG2* where only attributes within budget are considered and C4.5. The misclassification cost also reduces as the number of stochastic samples, r, increases, with the most significant improvement occurring when one, two or three samples are used, but minimal improvement after three samples.

3.3 Summary and analysis of the results of the survey

There has been significant interest in the introduction of costs into decision tree induction. Many ways of introducing costs within the decision tree process have been developed. Whilst there have been accounts of different types of costs, there has been no synthesis of the wide range of studies on cost-sensitive algorithms. An extensive survey of the field has been carried out with a view to providing an appreciation of the different approaches and algorithms that have been proposed. Additionally it provides a guideline to the type of framework which may prove effective.

---

[8] More formally, they select an i for which $(E_i - E_{i+1})\ (C_{i+1} - C_i)$ is maximum where the E_i and C_i are the expected error and budgets for the i^{th} tree.

A new taxonomy of cost-sensitive algorithms has been developed, organizing the algorithms into classes representing the way cost sensitivity has been introduced. The survey revealed two major approaches; greedy, which induces a single tree making decisions with no backtracking and non-greedy, which uses multiple trees and multiple choices to induce trees. Seven classes are defined:

1. *Use of costs during construction*, whereby attribute selection measures are adapted to include costs. The main differences between the algorithms in this class are the selection measures used and whether costs of tests, misclassification costs or both are incorporated.
2. *Post construction*, developed when costs are unknown at training time or if the costs are subject to many changes. Differences between these algorithms arise from how the labels for leaf nodes are chosen.
3. *GA methods*, which utilize evolution, producing populations of decision trees which are evaluated with regard to costs with the fittest being retained and combined. The algorithms vary in the way trees are generated or represented and how the fitness is measured.
4. *Boosting*, which generates a number of decision trees in sequence using instance weights. The algorithms differ in the way that these weights are initialized, and updated. Other differences between algorithms include how the sampling is done, and how error rates or confidence rates have been calculated in order to give the trees with least error more importance in composite voting methods.
5. *Bagging*, which generates a number of independent decision trees using re-samples from the training set, thus differing from the trees generated by boosting, being independent of each other similarly to those in the GA methods. Generally these algorithms are wrapping methods, using the decision tree as a sub-routine and wrapping the incorporation of costs

around it. Differences between these algorithms are how sampling occurs and in the composite voting method used.

6. *Multiple structures*, which expands the ideas of generating alternative trees and combining the outcome by having alternative trees in one structure. This shows all possible alternative choices of attribute selection in one decision tree so that alternative choices are not discarded as in the usual decision tree process but are stored and can be expanded in the future.
7. *Stochastic Approach,* which induces decision trees created by generating *r* stochastic samples of trees rooted at each potential attribute and selecting the attribute that results in the best tree. Varying the number of *r* samples results in the anytime behaviour where quality can improve with more time. As well as anytime behaviour, this approach has been used to produce a framework for anycost behaviour, where misclassification costs reduce as the available total cost for testing increases.

The survey also includes a timeline showing how the field has developed from early algorithms that simply amend selection measures to take account of costs, to the more recent and sophisticated stochastic algorithms that use sampling to induce anycost trees.

If one were to decide which of the existing algorithms may be appropriate for use in a particular domain, this would depend on various factors including whether an application needs to minimise costs alone, minimize costs of tests and misclassifications, whether there is a fixed budget for test costs, and whether there is a need for anytime or anycost learning. Although, the particular experimental methods, datasets utilized (see Appendix A1) and related systems compared vary, it is possible to form a general view from the empirical evaluations presented in the studies.

A number of the non-greedy algorithms show the benefits of generating multiple trees. Based on the original study by Turney (1995) and the independent comparisons in (Lomax and Vadera 2009, 2011), *ICET* performs well when aiming to minimize the sum of costs of misclassification and tests, especially when costs of misclassification are uniform[9].

ACT, a system based on stochastic sampling, improves upon the results from *ICET* for both uniform and non-uniform misclassification costs (Esmeir and Markovitch 2008).

The use of boosting has developed from the pioneering work on systems such as *Cost-UBoost* (Ting and Zheng 1998a) and *AdaCost* (Fan et. al, 1999) to *JOUS-Boost* (Mease et al. 2007) that shows the benefits of adding noise to the sampling process to reduce over-fitting. Lozano and Abe (2008) have advanced our understanding of cost-sensitive boosting by deriving methods such as *Cost-Sensitive Boosting with p-norm Loss* (L_p-*CSB*) that are guaranteed to converge. The recent work of Esmeir and Markovitch (2011) on *TATA* provides a novel framework for applications where the maximum cost for testing is available in advance, at the classification stage or even later.

Given the relative success of non-greedy algorithms for cost-sensitive tree induction, a fair question is:

> "Is it worth using or even pursuing future research on greedy cost-sensitive decision tree induction algorithms?"

The primary advantage of the greedy algorithms is that they are very efficient and therefore represent a good starting point for applications, and where performance with respect to costs

[9] Costs of misclassification are said to be uniform when they are the same for all the classes.

is very good, there may be little benefit in using the more computationally expensive multi-tree methods. Producing similar results to multi-tree methods using single tree methods does represent a major research challenge, but as the work on non-linear decision tree shows (Vadera 2010), it is possible to produce results comparable to *MetaCost* and *ICET* for minimisation of misclassification costs at a fraction of the computational time. Whether it is possible to extend this to applications that need to take account of costs of tests or budgeted learning remains an open question.

In conclusion, the field of cost-sensitive decision tree learning has a rich and diverse history, providing a strong base for future research which would include building upon recent advances to develop new algorithms that improve performance or meet new requirements

CHAPTER 4: THE DEVELOPMENT OF A NEW MULTI-ARMED BANDIT FRAMEWORK FOR COST-SENSITIVE DECISION TREE LEARNING

This chapter develops the framework for cost-sensitive decision tree induction which uses the principles of the multi-armed bandit algorithm. Section 4.1 presents a summary of the performance of previous algorithms which motivates the rationale, Section 4.2 develops the core algorithm and Section 4.3 identifies potential problems and some refinements. Section 4.4 summarizes the development stage.

4.1 Analysis of previous cost-sensitive decision tree algorithms

One of the first comparisons of cost-sensitive decision trees evaluated the Genetic Algorithm based system *ICET*, the use of Linear Machines (*LMDT*) and C4.5 (Vadera and Ventura 2001). A more comprehensive evaluation was carried out as part of an MSc (Lomax 2006) and later additional datasets were considered and the results published in Lomax and Vadera (2011).

Experiments were carried out over a range of cost matrices for the following algorithms: *EG2, CS-ID3, IDX, MetaCost, MetaCost_A, MetaCost_CSB, AdaCost, SSTBoost, CS-AdaBoost, CSB, LS-ID3, CS-LSID3, ICET*. A typical two-class cost matrix has a different value for misclassifying each class as the other, as below in Table 6. As another example, Table 5, given on page 28 is a cost matrix for a four class problem.

For example, to misclassify a class x example as class y, the misclassification cost would be 1, but to misclassify a class y example as class x has a misclassification cost of 100.

	Predicted class	
Actual class	x	y
x	0	1
y	100	0

Table 6 Typical cost matrix for two-class problems

Eleven datasets, some of which are available from the Machine Learning Repository, and coming from differing domains including medical diagnosis, construction, insurance and games, which were chosen for the differing characteristics and used in the comparison, are presented in Table 7 (Lomax 2006; Lomax and Vadera 2009, 2011).

data set	classes **(distribution in %)**	instances	attributes	type of attributes	test costs	type of test	grouped	discount
hepatitis	2 (83/17)	78	19	6 continuous	available	both	yes	available
tic-tac-toe	2 (65/35)	958	9	all nominal	all 1.0	immediate	no	all 1.0
heart disease	2 (54/46)	296	13	5 continuous	available	both	yes	available
house voting	2 (61/39)	435	16	all nominal	all 5.0	immediate	no	all 5.0
mushroom	2 (62/38)	300	22	all nominal	1.0-10.0	immediate	yes	1.0-10.0
chess	2 (53/47)	300	37	all nominal	all 1.0	immediate	no	all 1.0
coil	2 (80/20)	2500	21	all nominal	1.0-10.0	immediate	no	all 1.0
KDD98	2 (80/20)	5000	7	all nominal	1.0-10.0	immediate	no	all 1.0
MRI	2 (77/23)	400	15	all continuous	1.0-10.0	immediate	no	all 1.0
bridges	4 *	70	11	all nominal	1.0-7.0	both	yes	all 1.0
soybean	15 **	266	35	all nominal	5.0 or 10.0	both	yes	all 2.0

* 4-class distribution: 61/16/13/10%
**15-class distribution: 3 classes 15%, 2 classes 7%, remaining 4%

Table 7 Main characteristics of datasets used in the comparison

The result of this evaluation showed that to make a decision tree truly cost-sensitive, it is best to construct it using all the costs involved i.e., cost of the tests and the misclassification costs. Applying only misclassification costs after the decision tree has been constructed, for instance like the wrapper method algorithm *MetaCost*, to a certain extent does make the decision tree cost-sensitive, but can result in high classification costs when the cost of tests

are included in this calculation, owing to the fact that *MetaCost* as described, uses a weak learner which does not allow for this type of cost. Although the tree would be constructed with misclassification costs in mind, it may be that the attributes chosen to construct the tree have high test costs associated with them.

Likewise, using only misclassification costs in instance weighting, as used in boosting methods, makes the decision tree cost-sensitive in some ways, but does not always have the right effect on some of the classes in the dataset. One of the main observations from the empirical evaluation was that incorporating misclassification costs in this way seems to have the effect of sacrificing overall accuracy rates in order to produce lower classification costs (Lomax and Vadera 2009, 2011). Using only the cost of the tests does mean that even though the accuracy rate may not be high, the cost of classification would still be relatively low; however no allowance can be made for examples, which would have high misclassification costs. High accuracy rates do not always mean low classification costs, likewise having an inexpensive decision tree where the cost to classify an example is not high, does not automatically mean that it is an accurate decision tree.

It was noted in the conclusions to these experiments that the algorithm using the genetic method of incorporating costs, namely *ICET*, performed better overall, however studying the results of this 'best' algorithm highlight certain persistent problems in cost-sensitive decision tree learning which still have not been addressed.

The experimental methodology adopted by Lomax (2006) used a randomly generated training/testing split with 70% of the data for training and 30% of the data for testing. Ten

training and testing pairs were created. Measurements are averaged over these ten pairs. The measurements used to examine the performance are:

1. Average cost to classify an example (cost)

   The average cost of classifying an example is calculated using the test costs and misclassification costs as described by Turney (1995). An attribute (test) has a cost and may belong to a group. The first test is charged at full price, additional tests in the same group are charged at discount rate. Each time a new group is tested it is charged at full price.

2. Accuracy rates (accuracy)

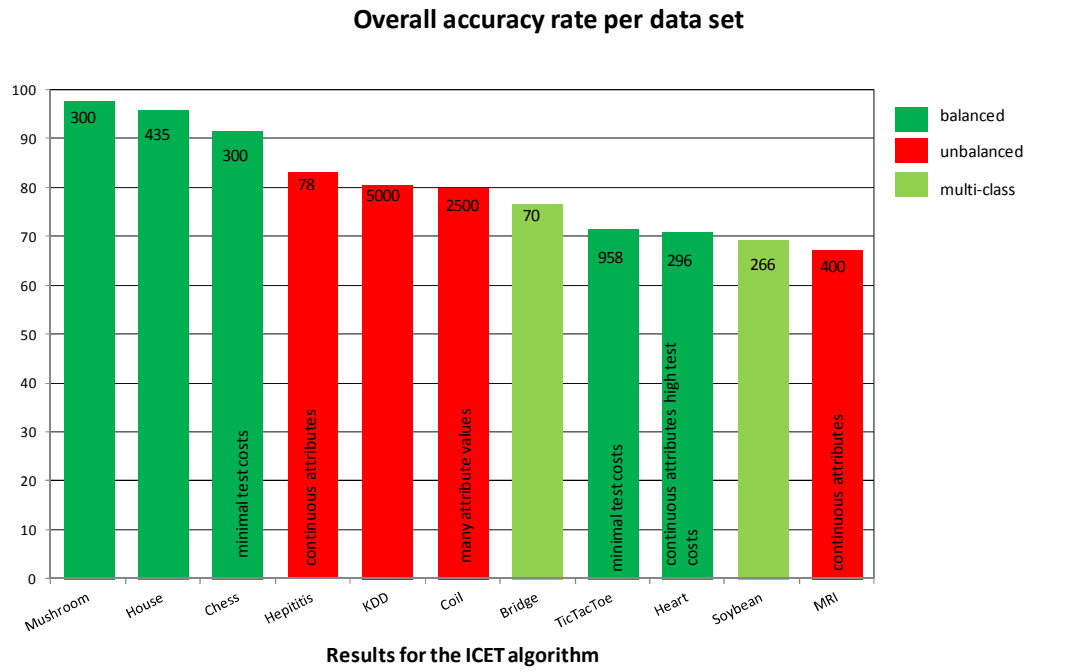
   The accuracy of the algorithms are measured by assessing how many of the examples in the testing set have been classified correctly as a percentage of the total number of examples in the testing set.

To find out the average cost to classify an example by each algorithm in each dataset, the test costs are used along with the relevant misclassification costs. Turney (1995) found that averaging costs for comparison was not suitable as test costs varied between datasets so he normalized the average cost by dividing by a standard cost using equation (4.1):

$$TC + min_i(1 - f_i)max_{ij}C_{ij} \quad (4.1)$$

where TC is the total of all tests, f_i is the number of class i examples divided by the total number of examples and C_{ij} is the highest cost in the current cost matrix.

This methodology has been used for every subsequent experiment described in this thesis.



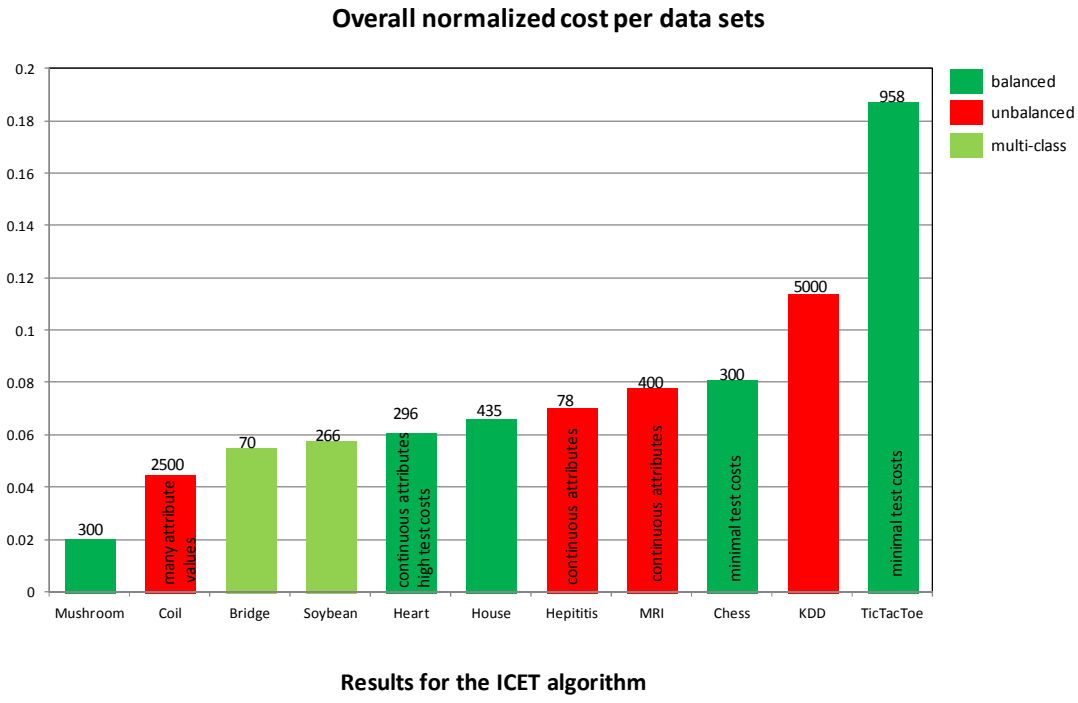


Figure 10 Using ICET's results to demonstrate weaknesses of cost-sensitive decision tree algorithms

Figure 10 shows the results of the *ICET* algorithm from the empirical evaluation, the accuracy rate and cost to classify are averaged over the cost matrices used, with particular attention to the structure of the datasets themselves. The numbers shown against each bar is the number of examples in the whole of the dataset. The datasets are ordered from left to right, best to worse results. It can be seen that for some datasets *ICET* does not return as high an accuracy rate as on others. Likewise the cost of classification can vary in the same way. It could be expected that the same performance would be returned for every dataset processed and expect the datasets to be presented in the same order in both graphs, but this is not the case. For example, the coil dataset returns the second best result for cost to classify but the accuracy rate returned is sixth best.

The best results for this algorithm are found on the datasets which are two-class and balanced. There are four datasets which are unbalanced; the one returning the worst result for accuracy belongs to the MRI dataset, but it does not return the worst cost to classify of these unbalanced datasets, some of the unbalanced ones produce a lower cost to classify than more balanced datasets. Two datasets which potentially should either present a problem to *ICET* or if not should return similar results, are those with minimal test costs. The chess dataset presents no problems to *ICET* which returns an accuracy rate of over 90% with a cost of 0.08 but the other dataset with similar test costs is tic-tac-toe which only returns an accuracy rate of 70% with a cost of 0.187.

Based on these results and the literature review, the following weaknesses in the cost-sensitive decision tree algorithm have been identified:

- Problems arise from an imbalance in the class distribution, with most decision tree learners biasing outcomes towards the dominant class; if this class is not the most costly, this explains the reduction of accuracy rates
- Multi-class datasets cause problems because the frequency of examples in each class may not be high making it difficult to distinguish between the classes; the classes themselves may be similar also. Additionally multi-class datasets have the characteristics of imbalanced datasets
- Extreme misclassification costs are difficult to handle since they result in bias and can result in no model being built; Lomax (2006) found that *MetaCost* returned no models when the misclassification cost range was high because the training set had all been labelled as the most costly class thus meeting the stopping criteria of decision tree algorithms with all examples in the one class
- Trade-offs between high misclassification costs usually result in the accuracy rate being sacrificed; the higher the misclassification costs, the more unbalanced the class distribution, the lower the accuracy rate

Even the 'best' algorithm (*ICET*) struggled with these aspects, for those algorithms which did not perform as well, these weaknesses were even more apparent. After studying the results presented in Figure 10 it is concluded that there is an additional factor to the class distribution, test costs and misclassification costs, which contribute to the way the datasets are processed.

The nature of datasets could account for these discrepancies and inconsistent performances although to what extent has not been investigated in the area of cost-sensitive induction. This has been investigated in the STATLOG project (King et al. 1995), who concludes that the dataset being investigated contributed to which algorithm produced the better results. In order

to determine which algorithm should be used they developed a way to describe datasets and recommend the best algorithm based on this description. The descriptions include whether there is a skewed distribution, correlation and the types of attributes in the dataset. Figure 10 includes descriptions of the datasets to indicate the main characteristic of that dataset. In addition to the observations in the STATLOG project, there are indications that the structure of the data, that is, the number of attributes, number of discrete values each attribute has and the number of classes, could influence how easy it would be for an algorithm to classify examples. Large test costs, minimal test costs and large misclassification costs also influence how well an algorithm processes the datasets. These observations are taken into account when choosing the datasets and allocating test costs and misclassification costs.

4.2 A new algorithm for cost-sensitive decision tree learning using multi-armed bandits

Use of costs within the decision tree learning process has introduced many interesting problems involving the trade-off required between accuracy and costs. It is clear that, whilst there are existing cost-sensitive decision tree algorithms which can solve two-class balanced problems well, other types of problems cause difficulties. It is suggested that there is still research to be done with regard to cost-sensitive decision tree learning in order to overcome more difficult and challenging problems of the kind identified in Section 4.1.

Evidence suggests that cost-sensitive learning needs to take account of the trade-off between decisions based on accuracy and decisions based on costs. For instance an algorithm using some sort of statistical measure might make a decision to split the data on a particular attribute as this is the desirable one based on an information theoretic measure. However, this attribute may have an expensive test cost associated with it. An algorithm employing only

costs may simply choose the least costly attribute, resulting in a different decision. If the aim is to minimize cost, a greedy algorithm will choose the cheapest one at every level. The ideal solution may be co-operation between these two viewpoints. A compromise could be reached whereby a slightly more expensive cost than the cheapest could be chosen so that a smaller tree is developed. This may reduce misclassification costs rather than simply using attributes in order of cost which may produce a large tree that does not reduce the misclassification costs as much as the other. High cost attributes may hide good splits (Esmeir and Markovitch 2008) however the reverse is also certainly true; that low cost attributes may hide bad splits.

Cost-sensitive learning could be thought of as involving two decision-makers because there is an algorithm and costs which sometimes work together well and sometimes do not. For example, player 1; 'accuracy-based player' chooses strategies concentrating only on accuracy and player 2; 'cost-based player' chooses strategies which consider costs in some way, each produces a different set of strategies. Conflict between decisions based on accuracy and decisions based on costs may produce good strategies but if they do not, a trade-off between these strategies must be found so a technique which deals in trade-offs should be utilized in a framework for cost-sensitive learning. What can be surmised at this stage is that the pay-offs matter when deciding strategies.

Multi-Armed Bandit problems are those which could be solved by trade-offs between exploration (trying out solutions or strategies to find the best one) and exploitation (using the solutions or strategies, which are believed to give the better payoff). The Multi-Armed Bandit Game, as described in Chapter 2, has been used for a variety of problems such as selecting routes for packages and allocation of money to different projects where the outcome is not known (Berry 1985, Gittings 1989, Auer et al. 1995; Auer et al. 2001, 2002; Vermorel and

Mohri 2005; Dorard et al. 2009; Grünewälder et al. 2010; Dorard and Shawe-Taylor 2010). In these applications trade-off occurs in order that total cost of sending a set of packets on selected routes would not be larger than sending the packets all together on a single route or the trade-off between potential research projects which may prove profitable but this information is only obtained over time. Given the trade-off between accuracy and costs, this thesis explores the use of multi-armed bandits for cost-sensitive decision tree learning.

The multi-armed bandit game is based on a gambler in a casino deciding which slot machine from a selection of slot machines is likely to pay out the most. The objective is to maximize the sum of rewards earned through a sequence of lever pulls. The player will randomly choose, from a given number of bandits, a lever to pull and will either get a reward or nothing. The player decides how many times he will explore the bandits by pulling levers a certain number of times, some bandits may be chosen more than once others not at all. The bandit which rewards the most will then be exploited by the player.

The bandit to be exploited is usually decided based on a measure of regret, the one minimizing this value is chosen. This is calculated as the difference between the reward sum associated with an optimal strategy and the sum of the collected rewards (Auer et al. 2002; Auer 2002; Shawe-Taylor 2010).

As it is suggested that cost-sensitive learning involves a trade-off between decisions based on accuracy and decisions based on costs, and as there is most definitely a trade-off between accuracy and lowering costs, the principles of the multi-armed bandit game could be used for decision tree learning to produce an algorithm which may address the problems indicated in Section 4.1.

A key step in decision tree learning is the selection of the attribute upon which to split the data. As detailed in Section 3.1.1.1, typically a statistical measure such as Information Gain is used. There are very few algorithms found in the survey which only use the costs to determine which attribute to use, detailed in Section 3.1.1.2. In this new approach, the multi-armed bandit algorithm will use both misclassification costs and test costs when selecting the attribute to exploit. The misclassification costs and test costs can be used as rewards; or rather the reduction of them is the reward. In order to gain as much information during the exploration stage as possible, a look-ahead methodology will be employed. A single bandit lever pull represents a test on an attribute. Figure 11 illustrates the idea. Given a set of attributes A, an attribute *a* is chosen at random. A value *v* belonging to attribute *a* is chosen at random followed by additional attributes and their values until the depth to look ahead is reached.

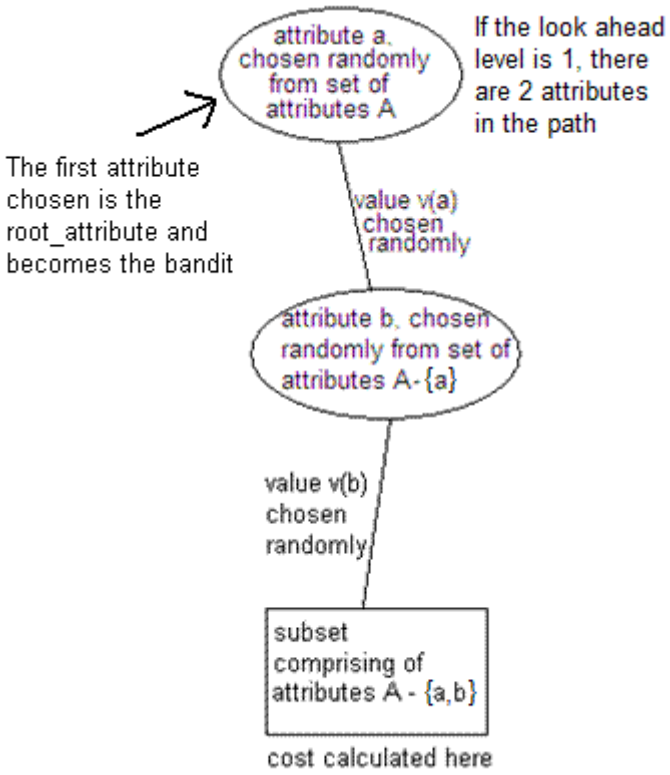


Figure 11 Illustration of the single pull and look-ahead bandit in the algorithm

The reward, or pay-off, is the reduction in costs, i.e. the ‘bandit’ (root_attribute), which reduces cost the most. As the multi-armed bandit algorithm sums up the rewards for each bandit, the average cost to classify an example will be summed up each time the same bandit is chosen. The average cost to classify is calculated using both misclassification costs and test costs.

Figure 12 illustrates P bandits which have been generated and the cost calculated at the end of the path. In the illustration there are five different bandits chosen; odor, sr, hab, gc and bruises. For each time they have been randomly selected, an attribute value has been chosen. In this example the look-ahead level has been set to 1, this means that there are 2 attributes in the path. This second attribute, along with one of its values has been chosen at random and the cost has been calculated. The cost is calculated by assigning a label to the end of the path, as described later in this section. Those examples which would then be misclassified have their misclassification costs summed up. For every example at the end of the path, the test cost is added to the total misclassification costs and then divided by the number of examples at the end of the path.

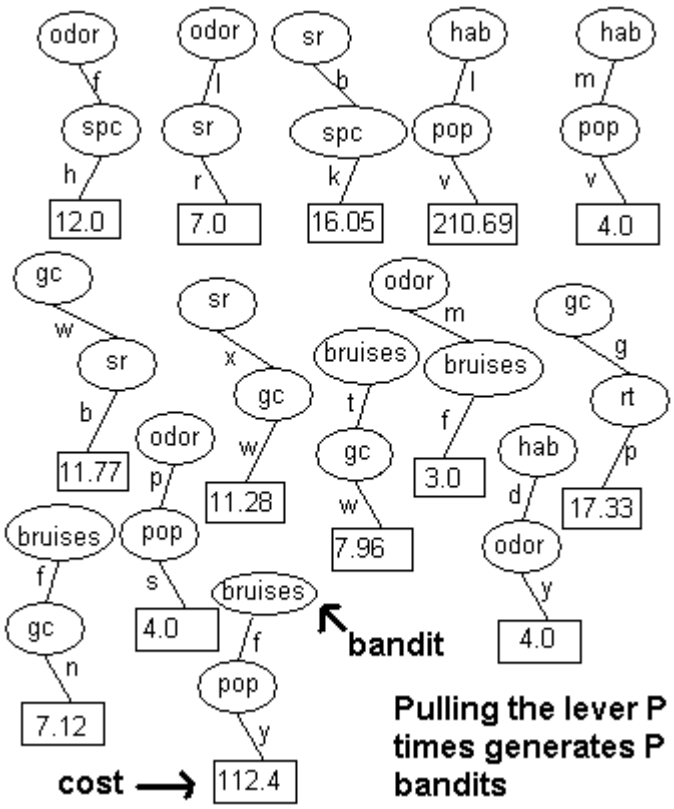


Figure 12 Generate P bandits and calculate cost at the end of each path

For example, the bandit 'odor' was chosen four times and the cost values 12.0, 7.0, 3.0 and 4.0. The mean value calculated is 6.5. Table 8 presents the cost values summed up for each bandit and the mean for each bandit. As the requirement is to reduce costs and to exploit the bandit which reduces costs the most, the optimal strategy will always be the lowest cost obtained after P lever pulls. The process can be simplified by examining the mean of the costs returned by the bandits and then choosing the bandit which obtains the lowest mean value. The selection process by the multi-armed bandit algorithm continues until it is not worthwhile to continue.[10]

[10] The issue of when to stop is discussed in Section 4.3

bandit	summed up cost	mean of costs
odor	26.0	6.5
sr	27.33	13.665
hab	218.69	72.89
gc	29.1	14.55
bruises	127.48	42.49

Table 8 Example of the multi-armed bandit algorithm choosing an attribute

Figure 13 shows the pseudo-code of the suggested framework. The multi-armed bandit algorithm is applied in the function calculate_mean_cost_for_A. The cost R_i is calculated and accumulated for attribute a_i. After the for-loop is executed, the attribute minimizing cost by obtaining the lowest mean cost value, is chosen for exploitation. Only those attributes which were selected for exploration are considered for exploitation. The function pull_lever_generate_path represents the lever pull of the multi-armed bandit. To exploit the attribute the data is split on the desired attribute.

To create a leaf, most existing algorithms either use the majority class or select the class that minimizes classification costs. However, in this algorithm a hybrid cost-sensitive labelling method has been employed, not seen in the literature. This labels the majority class regardless of cost but considers the cost when labelling those nodes where there is an even or almost even class distribution at the node. This is a combination of the labelling system of an 'accuracy-based player' algorithm and a 'cost-based player' algorithm, as the concept of this algorithm is to find a compromise between the two, perhaps having a labelling system which is a compromise between them might help.

```
Tree MA_CSDT(Examples, P, k)
/* Examples are the current dataset, P is the number of lever pulls
   and k is the depth to look ahead, subTree are the child nodes, leaf is
   an indicator that a leaf node has been created, classOfLeaf will
   contain the class of the leaf node */

   Initialize Tree subTree  // an array of sub-trees
   Initialize Boolean leaf as false
   Initialize classOfLeaf as null

   If not worthwhile to continue or no data left to explore
            assign classOfLeaf with the most appropriate class
            assign leaf as true
            return
   else
            attribute_to_exploit = exploreAttributes(Examples, P, k)
            subset = Split_Data(Examples, attribute_to_exploit)

            For each subset i
                  subTree_i = MA_CSDT(subset_i, P, k)
            END FOR

            return subTree  // the sub-trees
END

Attribute exploreAttributes(Examples, P, k)
/* R is an array corresponding to the set of attributes A and will
   contain the sum of costs obtained for an attribute at each lever
   pull, N is an array corresponding to the set of attributes A and
   will contain the number of times that attribute chosen as root of
   path, means is an array corresponding to the set of attributes
   A and will contain the mean of costs calculated for each attribute
*/

      Initialize R to zero
      Initialize N to zero
      Initialize means to zero

      For j = 1 to P
      Begin
            a_i ∈ set of attributes A
            Path_i = pull_lever_generate_path(a_i,A,k)
            R_i += cost for a_i (Path_i, Examples)

      End

      means = calculate_mean_cost_for_A(R,N)

      return attribute(minIndex(means))
END
```

Figure 13 Multi-Armed Cost-Sensitive Decision Tree Algorithm (MA_CSDT)

As mentioned above, most cost-sensitive algorithms label leaf nodes by selecting the class that minimizes cost of misclassification, whilst accuracy based algorithms typically select the majority class. However, when costs of misclassification in one class are significantly higher than another, there can be a tendency to label many leaf nodes with this class, effectively ignoring other classes. The above mentioned hybrid labelling system has been developed to try to avoid this. As an example, if there were 10 examples in the subset at the end of the generated path, with 8 in class 1 and 2 in class 2, with misclassification costs of 1.0 and 10.0 respectively, and with each attribute in the path having a test cost of 5.0. Using the hybrid labelling, the subset would be labelled as class 1. The examples in class 2 would incur 20.0 misclassification cost. Additionally there would be 100.0 test costs, thus the expected cost would be calculated as (20.0 + 100.0)/10 which is 12.0. If the cost-sensitive labelling were used, the subset would be labelled as class 2 because to get 8 class 1 examples wrong costs less, thus the expected cost would be calculated as (8.0 + 100.0)/10 which is 10.8. Although the aim in cost-sensitive learning is to reduce costs, the aim of this algorithm is to minimize the amount of sacrifice of the accuracy rate. To do this, the majority class should not be ignored completely. Ignoring the information in a subset, i.e. the evidence that these examples belong in the majority class regardless of cost does lead to the sacrifice of the accuracy rate. By compromising between labelling systems, it is thought that this will help find a compromise between the two opposing viewpoints.

The data mining software WEKA has been chosen to help in the development and implementation of the new algorithm. WEKA has been developed by Hall et al. (2009) using Java and has been used by a number of authors both to implement algorithms and in experiments.

4.3 Potential problems with the MA_CSDT algorithm

The algorithm developed in Section 4.2 has a number of open questions that need further consideration, including:

- What is a suitable choice for the number of lever pulls, P and look-ahead value *k*
- What is the best way to stabilize fluctuating results obtained when executing the algorithm repeatedly?
- Is it worth carrying on when there are only a few attributes or examples left in the dataset?
- Can the hybrid labelling system allow for the trade-off between accuracy and costs?

To investigate these questions, two artificial datasets were created using datasets from the Machine Learning Repository. Artificial datasets have been used so that the tree build can be anticipated and the outcomes controlled in order to examine development. The mushroom dataset was used to supply examples for 2-class problems, first with a balanced class distribution and then an imbalanced one and the glass dataset was used to supply examples for multi-class problems. A varying number of attributes chosen from the datasets were assigned test costs which would dictate choice in the tree induction.

The following sections address the above questions. Section 4.3.1 examines the problems of setting the value of P, Section 4.3.2 explores two techniques which could be used to stabilize fluctuating results, Section 4.3.3 reports the problems and solutions with regard to when to stop the process and Section 4.3.4 investigates the potential for using a hybrid labelling system. Finally Section 4.3.5 draws conclusions from the experiments carried out during the developmental stage.

4.3.1 Problems setting the number of lever pulls and depth to look ahead

If there is not enough exploration, potentially not enough different bandits are chosen which can result in an expensive attribute being chosen too early. An attribute with a high test cost can be chosen as a root attribute, thus returning a higher cost to classify than anticipated simply because the dataset space has not been explored enough. To make sure this scenario does not arise there must be a sufficient number of lever pulls carried out. If the number of lever pulls is high enough the algorithm has the opportunity to explore many more potentially good attributes. A way must be found to determine, what is a 'sufficient number'. To determine what value is best for the look-ahead parameter *k*, a dataset was processed with increasing values of *k*. On examining the results obtained, it was found that there was no improvement in the results obtained at each increase of depth. It has been decided that further experimentation would be required as this result may be unique to this dataset. This parameter will therefore be set to the value 1 for all datasets and set to 2 for a selection of datasets to determine whether a lower depth improves results or not and if it is dataset related.

It is evident that just guessing on the ideal value for the number of lever pulls for a given dataset is not good enough. Perhaps there is some way to calculate this? If the maximum number of unique bandits there would be in a dataset could be estimated in some way and this is used as a guide in the setting of the value of P, more stable results may be produced. To illustrate the problem of calculating the potential number of unique bandit paths which could be used in the multi-armed bandit algorithm, suppose there is a dataset with three attributes and each of these attributes has two values. If the depth has been assigned to 1, there will be two attributes in the path, and these attributes must be different. The solution to this problem can be drawn out as presented in Figure 14, where all potential unique bandits for the dataset are illustrated.

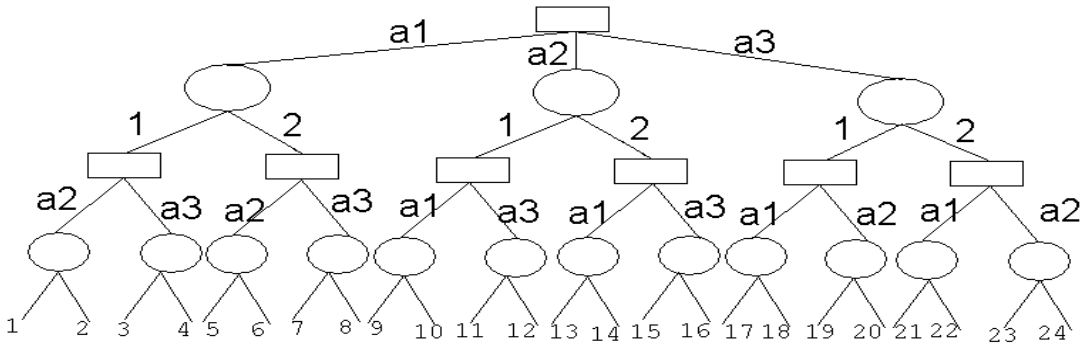


Figure 14 To calculate the number of potential unique bandit paths in a dataset

It is now simply a case of counting out the leaves, so in this problem the solution is that there are 24 potential unique bandit paths which could appear in a decision tree. This very simple problem quickly becomes complicated. For a typical dataset, it is more problematic to draw out. The mushroom dataset, for example has 21 attributes. These attributes have different numbers of values ranging from 2 to 12. As any of the attributes can be a potential root, with any of its values in the path, it is not just a simple calculation and in order to calculate potential unique bandit paths it is necessary to enumerate out the calculation. To emulate the method illustrated in Figure 14, which is the easiest way to calculate the value required, a computer program which generates in turn every potential unique bandit path down to the depth of look ahead required has been used. Each time the end of a path is reached, a count is incremented. Thus the maximum value is then returned. This has been performed on the entire dataset but it could potentially be performed on a training set as it does not require knowledge of the number of examples, simply the attributes and their values.

The potential unique bandit path value for the mushroom dataset has been calculated as 12,624. Experiments carried out with the mushroom dataset then used a more realistic value of P as 10,000 which produced more stable results. Thus the value of P will be unique to each

dataset and will be assigned with the value of the potential unique bandit paths for that dataset as a guideline.

4.3.2 How to stabilize fluctuating results obtained during the developmental stage

As experiments on the artificial datasets were repeated, the results fluctuated, for example, sometimes the cost to classify obtained would be small but then on repeating the experiment the cost to classify obtained would be higher, so how could the results be stabilized? When a training and testing pair was subjected to repeated processing, the results obtained showed that there were differing trees resulting in high costs as well as low costs.

Figure 15 illustrates the results obtained when running the algorithm on the same dataset with the same costs ten times for each value of lever pulls used. This graph is representative of all variations of the artificial datasets used whilst developing the algorithm. In these experiments the number of lever pulls had been set to values between 50 and 300. It was concluded that this value was not high enough to guarantee sufficient exploration in that the number of different bandits chosen was too limited.

There are two potential ways to produce stable results. It may be that as described above, simply finding the ideal value of P will correct this problem. However, as noted in the literature study, constructing multiple trees has been a way to improve on the results obtained by using single trees. Therefore inducing multiple trees and then choosing the ‘best’ one might be a better solution. Game Theory involves a collection of models, where the one meeting the strategy is used. The main strategy of this algorithm is to lower costs whilst maintaining the accuracy produced by the ‘accuracy-based player’ algorithm. So therefore a number of trees can be generated using the above described algorithm, that is, if one tree is

generated the single version of the MA_CSDT algorithm can be used; if more than one tree is generated a multiple-model version of MA_CSDT can be used.

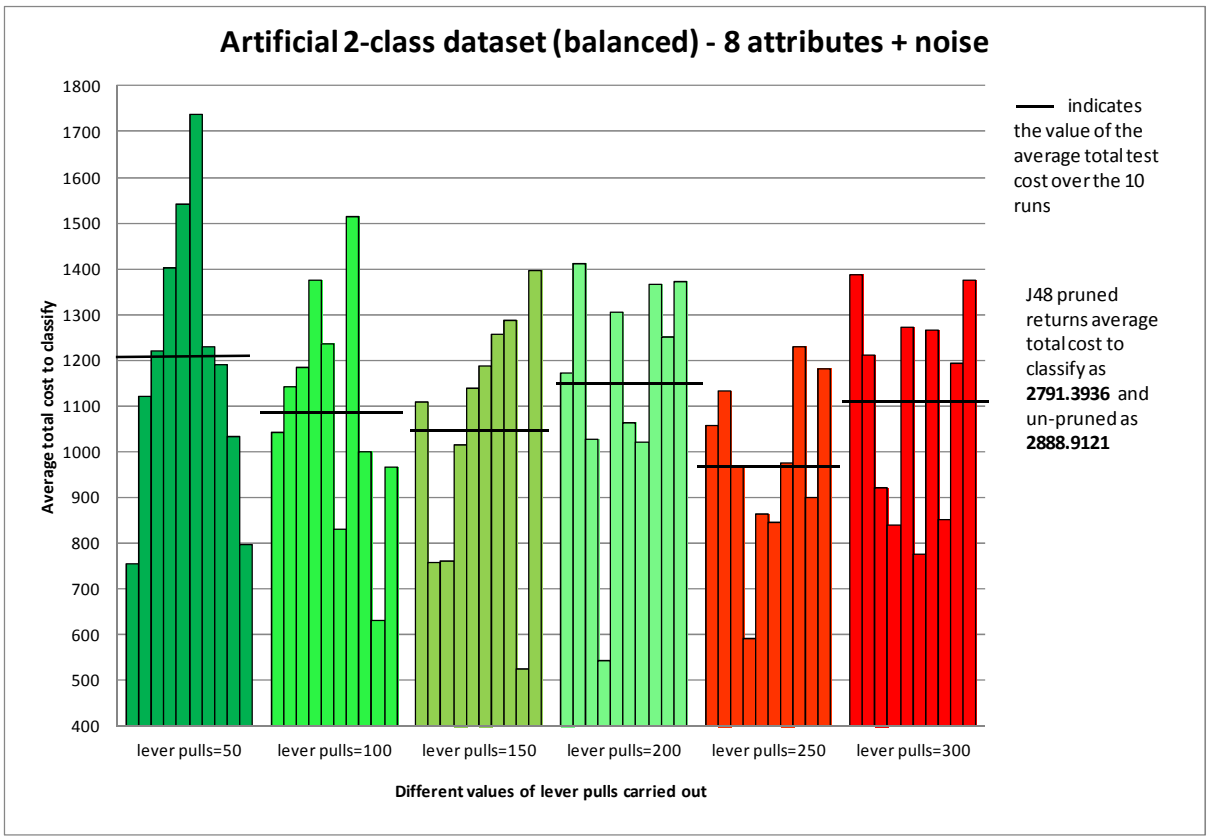


Figure 15 Results illustrating the fluctuating values obtained using artificial datasets

What should be the criteria for choosing the 'best' tree? Choosing the tree with the lowest cost seems to be the obvious choice, however this may not necessarily be an accurate tree as observed in previous comparisons. Choosing the tree with the highest accuracy is the other alternative but again this may not necessarily be a low cost tree, if the test costs are higher than the misclassification costs. The relationship between test costs and misclassification costs can be explored if both strategies are implemented and compared. Therefore the strategy can be set to choose the model which obtains the lowest cost or set to choose the model which obtains the highest accuracy. Given the strategy and the number of trees to be induced, the trees are evaluated on the training set to find out the accuracy of each tree and

the average total cost of each tree. Then, based on the strategy chosen, whichever tree meets the strategy will be returned to be evaluated on the test set. This idea is illustrated in Figure 16. The number of trees to induce has been set to a default of 10 which is the chosen default number of *MetaCost*, however this can be changed as desired.

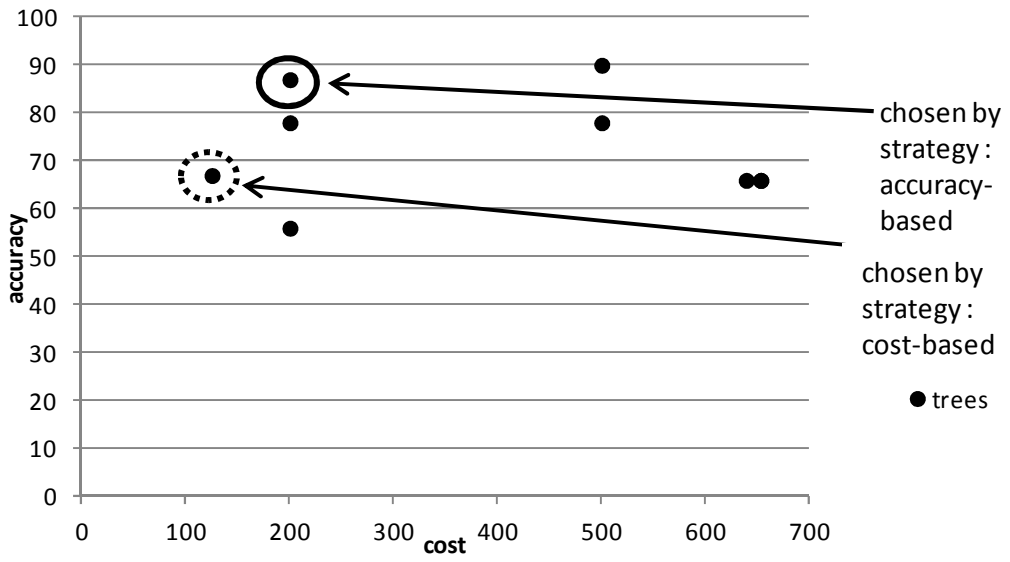


Figure 16 Desired tree chosen by a given strategy

For simple datasets it is possible that any strategy would result in the same model but for difficult datasets it would most likely result in differing ones and this is another way in which improved results can be obtained.

4.3.3 To examine when to stop the decision tree build process

Is it worth carrying on when there are few attributes left in the dataset which may be expensive? At some point particularly in the latter stages of decision tree build, the number of attributes is sufficiently reduced to limit random choice of attributes all of which may have high test costs. A pre-pruning technique is indicated given the nature of the multi-armed

bandit algorithm; exploring and gathering of information in order to find the correct bandit to exploit. Stopping when it is no longer worthwhile and there is no advantage to be gained by continuing is a logical step in the development of the algorithm. It also became clear that pre-pruning techniques would be necessary when exploring the relationship between test costs and misclassification costs and that pre-pruning techniques would be useful in order to find a trade-off between the two types of costs. With regard to expenditure of test costs, a solution would be to implement a test to determine whether the attribute chosen should be used. To check if the tree building process is worth continuing, the same test can be used at the start of the process. This test will determine whether the reduction of misclassification costs is less than the test cost expenditure.[11] To investigate which part of the tree build process will benefit more from this kind of test, the algorithm can employ these tests in combination.

Additionally, is it worth carrying on with processing if only a small proportion of examples are left in the dataset, how much information can be obtained in this situation? In decision tree literature there have been various studies regarding when to stop the procedure. These include stopping when the number of examples fall below a given number and then assigning the majority class as the label (Weiss and Kulikowski 1991) or measuring a split in some way and if this falls below a specified threshold stopping the process (Han and Kamber 2001). These measures aim to avoid the decision tree containing nodes with few examples which results in insignificant statistical measures with the aim of improving the accuracy of the decision tree. How should these values be determined? For larger datasets 'a few examples' could number in the hundreds but for a smaller dataset this value may only be five or ten. Using proportional measurements will be able to resolve the issue of what is 'a few'. The proportion of the dataset at a node in relation to the size of the original training set can be

[11] Ling et al. (2004) also uses a similar approach, except they only perform the test at the beginning of the process testing all attributes to determine whether to create a leaf node or continue with the tree build

calculated and, given a desired proportion, in anticipation that it would not be worth processing further, and stop when this proportion has been reached; the default value has been set at 5% or less. This value was chosen because it is low enough for the information to be exhausted from the dataset but not low enough that it would have no significant effect. It is necessary to be able to observe any effects. Also the proportion of majority class at a node can be calculated and, given a desired proportion, to determine whether to make a leaf node; the default value has been set at 90%. This value has been chosen as 90% since it is considered to be evidence enough that the class with this proportion is most likely to be the class for that particular node.

As the attribute chosen to be exploited is based on cost, it is necessary to make sure that this attribute has more than one value as using only costs will not give an indication of having only one value as a statistical measure would, hence a parameter is introduced to control what happens if this situation arises. If the chosen attribute has only one value, there are three options; (i) stop the process, (ii) ignore and continue with chosen one and (iii) choose the next best attribute or best one with more than one value. It is clear that ignoring and continuing to use a redundant attribute is not cost-effective but it is the intention to examine all possibilities to demonstrate this intuition.

4.3.4 Can the hybrid labelling system allow for the trade-off between accuracy and costs?

Does the hybrid labelling system allow for the trade-off between accuracy and the costs as it is thought it will? In past comparisons (Lomax and Vadera 2009, 2011), it has been noted that in order to achieve low costs, the accuracy rate is often sacrificed. The aim is to include information obtained from the majority class as well as the minority classes in order to

minimize the sacrifice of the accuracy rate but still lower costs. In order to test the hypothesis, and to determine whether this approach is successful, experiments will be carried out using both the hybrid labelling system and a cost-sensitive labelling system to be able to examine what happens to the accuracy rate.

4.3.5 Conclusions drawn from the developmental stage

Following the experiments carried out whilst developing the algorithm and aiding in identifying problems and possible solutions, small scale trials were carried out in order to gauge progress of the MA_CSDT algorithm. It became clear that 'one size does not fit all' in that it could be difficult to process a dataset with differing cost matrices, but by having parameters which can change behaviour and adjust to the differing characteristics of datasets can help with the process. For example some parameter settings may produce good results for a particular cost matrix but these parameter settings may not produce good results for a different cost matrix. Other different parameter settings can be examined to find good results so that there can be different parameter settings used for each cost matrix if required. Both the hybrid labelling system and a cost-sensitive labelling system, where the class which minimizes cost is allocated, were used in these small trials with the hybrid labelling system producing more high accuracy results than the cost-sensitive system.

These trials in the developmental stages used both versions of the algorithms, both of which could produce good results. Setting the value of P high enough helps, along with choosing from multiple models. But which is better over different datasets? It may be the case that some datasets are easier to process and do not require multiple models, but others require multiple models with a careful decision of the strategy required. Can guidelines be produced in order that parameter settings can be identified, which would also include whether multiple

models are needed or not, and which is the best strategy given the costs for a particular dataset? These questions are explored in the next chapter.

4.4 Summary of the development of the MA_CSDT algorithm

In the first instance, experiments from previous studies were examined in order to identify possible weaknesses in cost-sensitive decision tree learning. Analysis showed that to make a decision tree truly cost-sensitive it is better to construct it using all costs involved i.e. misclassification costs and test costs. Studying the results of the *ICET* algorithm revealed persistent problems in cost-sensitive learning which have not been addressed. These are (i) problems arising from class imbalance, (ii) multi-class problems, (iii) extreme misclassification costs and (iv) trade-offs between high misclassification costs resulting in the accuracy rate being sacrificed.

It is suggested that cost-sensitive learning involves trade-off between decisions based on accuracy and decisions based on costs and between accuracy and lowering costs so using the multi-armed bandit game which explores possible strategies before exploiting the best one, can be used with decision tree learning in order to produce an algorithm which may address the above issues.

To represent the lever pulls of the multi-armed bandit, a path containing two or more attributes chosen at random will be generated and the average cost to classify an example travelling down that path will be calculated. A hybrid labelling system has been developed which will take into account the information obtained from the class distribution at the node before deciding which class to choose.

The algorithm has a number of parameters, including the number of lever pulls to generate bandits P and the depth to look-ahead *k* which could affect performance on datasets in that these would increase or decrease exploration. In addition, based on the literature and past studies, additional parameters were introduced to explore (i) when the learning process should stop (ii) whether multiple models can be generated in order to obtain better results, and (iii) choosing a model returning the lowest cost (cost-based strategy) or choosing a model returning the highest accuracy (accuracy-based strategy). It is thought that by using these different combinations of parameter settings, the behaviour of the algorithm can be changed to suit the dataset being processed, the test costs assigned to it and the misclassification costs.

Two small scale trials were conducted to gauge progress in the development of the algorithm and used the full versions of the datasets which had been used to create artificial datasets. The conclusion of these trials is that 'one size does not fit all' in that choosing the most appropriate parameter settings for an individual cost matrix can produce good results and improve upon the overall results when just considering the same parameter settings for all cost matrices for a dataset. It was determined that the hybrid labelling system did increase accuracy rates and should therefore be investigated further to see if it helps with the trade-off between accuracy and costs.

In the following chapter, an extensive investigation has been carried out using 15 different datasets, which have been allocated varying values of test costs and misclassification costs in order to further address these issues.

CHAPTER 5: INVESTIGATING PARAMETER SETTINGS FOR MA_CSDT

An investigation has been carried out using 15 real-world datasets, which have a variety of different values of test costs and misclassification costs. Experiments have been devised in order to provide statistics with which to examine performance, using both versions of the algorithm; single-model version and multiple-model version using a cost-based strategy and an accuracy-based strategy, with all possible parameter settings using a variety of differing lever pull values for each dataset. Four of the datasets, which have a relatively small value for their potential unique bandit path value, were chosen to be processed to a lower depth of 2.

The main aim of the investigation is to help decide:

- Which parameters allow the algorithm to only continue when it is worthwhile to do so and what effect this has on cost and accuracy
- How many lever pulls should be used for each dataset, which version of the algorithm is better if any and which strategy for the multiple-model version works best for each dataset?
- Investigate how to take advantage of these different parameter settings and strategies in order to achieve the aim of the algorithm
- Can guidelines be devised for a dataset with regard to the combination of these parameters?

The datasets have been chosen to cover a range of characteristics, i.e. combinations of different numbers of attributes and values, number of classes and different class distributions. Some of these datasets have been used with test costs in previous studies, so the same test

costs have been used, making sure they have a good contrast of costs. The datasets are all available from the Machine Learning Repository but with some dataset files being obtained from previous studies. Full details of these datasets are in the Appendix A2 along with test costs, group information and discounted cost if any. The same experimental methodology detailed in Section 4.1, has been carried out on the 15 datasets.

A range of misclassification costs were used in order to examine the trade-off between the two types of cost used. For example misclassification costs were assigned to the classes in a dataset to be either higher than the test costs; lower than the test costs or a mixture of high and low values in relation to the test costs. Details of all cost matrices used in the experiments are listed in the Appendix A3. An identification number has been allocated to each one in order that they can be easily identified in tables and graphs.

Potential unique bandit paths have been calculated for each dataset, unique to that dataset, and used as a guideline to allocate values so that there are 5 values for each dataset. These values are (1) lower than the potential unique bandit path value; (2) rounded down from the potential unique bandit path value, (3) the potential unique bandit path value, (4) higher than the potential unique bandit path value and (5) a much lower value than any of the previous values. The identifiers 6 and 7 were specially included for soybean and used in one smaller experiment. This was done as the soybean dataset has a longer process time than all other dataset. The values for each dataset are presented in Table 9.

	Identification of the value of P for each dataset						
depth (*k*) 1	1	2	3	4	5	6	7
dataset							
annealing	7000	8000	8312	10000	1000		
breast	1500	2000	2152	5000	500		
car	200	300	366	500	50		
diabetes	150	180	184	300	30		
flare	700	800	898	2000	100		
glass	200	300	338	1000	50		
heart	500	600	658	1000	100		
hepatitis	800	1000	1082	1500	100		
iris	75	100	108	500	30		
krk	1000	1300	1312	1500	100		
mushroom	10000	12000	12624	15000	1000		
nursery	500	600	632	1000	100		
soybean	1000	3000	9104	5000	50	8000	9000
tic-tac-toe	500	600	648	1000	100		
wine	1000	1200	1256	1500	100		
depth (*k*) 2							
dataset							
car	3000	4000	5082	6000			
diabetes	500	1000	1776	3000			
glass	3000	4000	4692	6000			
iris	500	600	648	1000			

Table 9 Values of P lever pulls for each dataset

Experiments were designed so that every possible combination of parameter setting was executed for all training and testing pairs, and for each of the cost matrices described. As a result of the experiments executing all possible parameter settings, there is a large amount of data to be processed. After averaging out the ten training and testing pairs for each dataset, there are 158664 individual results to examine. To make it easier to find patterns and useful information all the results have been compiled into a dataset and used as input to the statistical software package SPSS in order that analysis can be performed on it. The table in Appendix Section A4 presents all the information which is contained in this results dataset and Appendix D9 is the comma separated value file containing all examples.

Each example in the results dataset has been classified by the value of the cost and accuracy returned by that experiment. If the cost is less than 0.3, the category allocated is ‘good’; less

than 0.5 is ‘medium’ and ‘poor’ otherwise. If the accuracy is greater than 68.0, the category allocated is ‘good’; between 45.0 and 68.0 is ‘medium’ and ‘poor’ otherwise. The values of these two attributes determine the class, based on accuracy/cost; (1) good/good, (2) good/medium, (3) good/poor, (4) medium/good, (5) medium/medium, (6) medium/poor, (7) poor/good, (8) poor/medium, (9) poor/poor. As combinations of accuracy and cost would result in a large number of classes, it was decided that three for each measurement would be more advisable rather than four each as the resulting combination would mean too many classes. Splitting each value into equal thirds was rejected as this would produce a ‘good’ accuracy of 66% or above. This was not considered to be high enough so the boundary was raised to 68% or above. A medium result of 33% and above was also considered too low to be in what potentially could be a reasonable result so this was also raised to 45% and above. Similar considerations to boundaries on the cost measurement was also given, thus less than 0.3 can be considered ‘good’ and higher than 0.5 is considered ‘poor’. Analysis performed on the dataset included cross-tabulation, frequency information and mean calculations obtained for cost and accuracy on attributes and their values in relation to the class allocated to examples.

The details of the experiments with their parameter settings are presented in Table 10. The table gives information regarding the strategy used; either no strategy because a single model is induced, cost-based strategy where from multiple models the one returning the lowest cost is chosen and accuracy-based strategy where the one returning the highest accuracy is chosen. The number of models, the depth and labelling system employed in each experiment are also shown in the table. Additionally frequency of best and worst results as a percentage, which are calculated using the previously mentioned class allocations and the frequency a model was produced by each experiment are also given in the table. For example the information

obtained from row 1 indicates that Experiment 0 has no strategy as only 1 model is induced. The depth to look-ahead is set to 1 and uses the hybrid labelling system. So for example for the annealing dataset, four different values of P were assigned. All the remaining parameter settings are used in combination which results in 96 batches of settings to be executed on the ten training and testing pairs for each cost matrix and dataset.

expId	strategy used	no of models	depth (*k*)	label system	best all examples (trees)	worst all examples (trees)	frequency tree produced	batches per dataset
0	none	1	1	HYBRID	37.8 (38.4)	1.51 (2.4)	50.7	96
0l	none	1	1	COST	29 (36.9)	1.36 (2.8)	38.7	96
0T5	none	1	1	HYBRID	34.8 (34.4)	1.58 (2.4)	51.8	24
0lT5	none	1	1	COST	26.9 (33.3)	1.34 (2.7)	40	24
1	accuracy	10	1	HYBRID	44.5 (52.4)	1.51 (2.4)	52	96
2	cost	10	1	HYBRID	39.4 (43.1)	1.49 (2.5)	48.3	96
3	accuracy	10	1	COST	34.8 (47.9)	1.45 (2.8)	41.7	96
4	cost	10	1	COST	29.9 (37.6)	1.42 (3.1)	37	96
5	accuracy	10	2	HYBRID	32 (26.8)	0 (0)	57.4	96
6	cost	10	2	HYBRID	28.2 (20.2)	0 (0)	57.4	96
7	accuracy	10	2	COST	19.1 (18.7)	0 (0)	43	96
8	cost	10	2	COST	18 (16.2)	0.019 (0.044)	43	96
9	accuracy	20	1	HYBRID	22.2 (37.5)	0.27 (0.46)	59.1	24
10	accuracy	10	1	HYBRID	0 (0)	7.87 (9.6)	45	96
13	accuracy	10	1	COST	77.5 (77.5)	0 (0)	100	8
S1T5	accuracy	10	1	HYBRID	39.9 (44.4)	1.52 (2.43)	52.8	24
S1lT5	accuracy	10	1	COST	30 (38.7)	1.36 (2.6)	43.4	24

Every combination of parameters as described in Section 4.3, each one of the combinations is executed on each of the datasets and their cost matrices.

Table 10 Details of each experiment, settings and frequency (%) of best and worst results from the analysis file

Table 11 presents a summary of the main information obtained by running analysis on the results dataset. For each dataset, the number of attributes is recorded, along with how many values the attribute with the highest number of values has. The potential unique bandit paths have been calculated to a depth of 1. A ratio of the highest misclassification cost divided by the highest test cost for each dataset and for each cost matrix has been calculated in the

results dataset, and the mean for each dataset for this ratio is recorded. These help to describe the type of dataset. Also included are the identifiers of the best experiments and best parameter settings producing the majority of best results in the order of (i) strategy chosen if 'best' attribute has only one value, (ii) an indicator of how many lever pulls carried out, (iii) the pre-pruning combination used and (iv) whether proportional stopping used; t if used f otherwise. If an 'x' is present in the group of parameter settings, this indicates that there is no overall parameter setting producing the best results.

For example for the annealing dataset, examining the best results obtained using the cost-based strategy, the experiment which produced the majority of the lowest costs was Experiment 4, which uses the cost-based strategy in order to choose a model, using a look-ahead depth of 1 and using the cost-based labelling system. The parameter settings, which produced the majority of the lowest cost, where the value 0 indicating what to do when the attribute has only one value for the best attribute, in this case stopping the process if this occurs; PId 4 which indicates the lever pull value used was higher than the potential unique bandit path value and in this case was set at 10,000 as shown in Table 9; the pre-pruning combination 'd' which does not use any checks to see if it is worthwhile to continue and 't' which indicates that the proportion of examples at a node would be calculated and the process would stop when this falls below the default level. For the accuracy-based strategy, there was no overall experiment producing the majority of best results, there would have been no checks for whether the 'best' attribute had only one value and no overall clear majority for a value of P. However the pre-pruning combination 'd' was again the better one most often but in this case there were no tests in order to carry out proportional stopping.

Dataset Name	annealing	breast	car	diabetes	flare	glass	heart	hepatitis
area	physical	medical	social	medical	physical	forensic	medical	medical
no of attributes	24	9	6	6	10	7	11	16
highest no. of values for an attribute	10	13	4	4	6	4	4	3
potPaths	8312	2152	366	184	898	338	658	1082
class distribution	unbalanced	unbalanced	unbalanced	65/35	unbalanced	part	even	unbalanced
multiclass	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE
group cost	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
mean ratio mc/tc	2.55	23.88	2.77	97.51	1.24	6.26	21.58	267.63
max(P)	10000	5000.00	6000.00	3000.00	2000.00	6000.00	1000.00	1500.00
min(P)	1000	500.00	50.00	30.00	100.00	50.00	100.00	100.00
dn_cost mean	0.027	0.08	0.09	0.15	0.02	0.14	0.00	0.14
cost (examples) mean	0.024	0.14	0.16	0.12	0.04	0.19	0.04	0.10
cost(trees) mean	0.0278	0.38	0.33	0.27	0.11	0.25	0.15	0.23
dn_accuracy mean	73.99	61.07	62.05	58.05	79.60	32.37	50.25	63.63
accuracy (examples) mean	82.07	60.17	63.21	60.35	82.81	44.99	55.78	69.78
accuracy (trees) mean	86.74	63.16	61.40	66.57	85.39	51.61	72.23	79.70
% of best results (examples)	97.7	48.90	68.50	11.00	89.10	2.38	19.40	70.30
% of best results (trees)	96.5	4.22	21.50	25.90	90.00	3.33	77.60	71.20
% of worst results (examples)	0	0.00	0.00	0.02	0.00	0.00	0.00	0.00
% of worst results (trees)	0	0.00	0.00	0.04	0.00	0.00	0.00	0.00
% whether a tree was grown or not	57	34.90	28.10	42.50	25.00	71.40	25.00	42.30
best experiment (strategy0)	4	4	4	x	x	x	4	4
best experiment (strategy1)	x	x	x	5	x	1	1	1
best parameter settings (strategy0)	0,4,d,t	0,1,d,t	0,4,d,t	0,1,d,t	0,1,d,t	2,4,b,t	0,4,d,t	0,4,d,t
best parameter settings (strategy1)	1,x,d,f	0,x,d,t	1,x,d,f	2,x,d,t	0,2,d,t	0,1,d,f	2,x,d,f	x,x,d,t
average time in seconds (single)	3.47	0.61	0.15	0.06	0.21	0.09	0.21	0.31
average time in seconds (ensemble)	31.56	4.69	1.33	0.35	1.76	0.64	1.89	2.60

Dataset Name	iris	krk	mushroom	nursery	soybean	tictactoe	wine	averages/ overall highest, lowest etc
area	botanical	game	botanical	social	botanical	game	physical	
no of attributes	4	6	21	8	35	9	13	
highest no. of values for an attribute	3	8	10	5	7	3	4	
potPaths	108	1312	12624	632	9104	648	1256	
class distribution	even	almost	even	even	almost	65/35	even	
multiclass	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	
group costs	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	
mean ratio mc/tc	1.22	9.51	317.34	4.73	70.00	2221.40	1.21	195.86
max(P)	1000.00	13000.00	15000.00	1000.00	9104.00	1000.00	1500.00	15000.00
min(P)	30.00	100.00	1000.00	100.00	50.00	100.00	100.00	30.00
dn_cost mean	0.11	0.36	0.00	0.19	0.32	0.20	0.10	0.14
cost (examples) mean	0.15	0.39	0.01	0.19	0.14	0.11	0.08	0.14
cost(trees) mean	0.29	0.43	0.02	0.26	0.14	0.24	0.15	0.23
dn_accuracy mean	33.33	15.22	50.11	38.36	16.31	57.44	34.76	47.46
accuracy (examples) mean	60.42	16.60	62.33	49.90	65.87	62.19	49.99	56.85
accuracy (trees) mean	86.29	18.76	99.01	61.75	65.88	69.73	82.85	64.86
% of best results (examples)	29.50	0.00	25.00	9.90	55.30	18.60	29.10	33.70
% of best results (trees)	57.80	0.00	100.00	19.50	55.30	43.10	87.70	38.90
% of worst results (examples)	0.00	14.90	0.00	0.46	0.00	0.00	0.00	1.30
% of worst results (trees)	0.00	28.50	0.00	0.92	0.00	0.00	0.00	2.30
% whether a tree was grown or not	51.10	41.70	25.00	50.80	99.90	43.10	33.10	46.00
best experiment (strategy0)	s1IT5	4	0	4	1	4	4	
best experiment (strategy1)	x	1	s1T5	x	1	x	x	
best parameter settings (strategy0)	0,5,a,t	0,1,b,t	2,1,d,t	x,4,d,t	x,3,c,f	0,2,d,t	0,4,d,t	
best parameter settings (strategy1)	2,x,d,f	1,5,d,t	2,5,d,f	0,5,d,f	1,3,c,f	2,4,d,f	2,5,d,f	
average time in seconds (single)	0.03	3.39	5.21	1.51	6.71	0.38	0.18	1.50
average time in seconds (ensemble)	0.10	40.71	51.06	13.46	64.15	2.90	1.59	14.59

Total no of experiments	158664
Total no of experiments producing trees	73024
Range of attribute numbers	4 - 35
Range of maximum number of attribute values	3 - 13
Range of potential paths	108 - 12624

Table 11 Summary of dataset information including mean values obtained from the analysis file

The annealing dataset has the highest frequency of best results and the krk dataset has the lowest frequency of best results when considering all examples. When considering only those examples when a tree is grown, the krk still remains the one with the lowest frequency of best results but the mushroom dataset now has the highest frequency of best results at 99%. The dataset with the most trees grown is soybean and with the least grown are flare, heart and mushroom datasets.

The following sections describe the findings of the investigation, Section 5.1 examines the parameters which ultimately control tree growth by permitting the process to continue only when it is worthwhile to do so and examine what happens to the cost and accuracy rate when each of the parameters are in control. Section 5.2 investigates the parameter settings which indicate how many lever pulls are being carried out, which version of the algorithm is better for certain datasets and the strategy which may be better, given the test costs and the misclassification costs assigned to each dataset. Section 5.3 investigates how choosing the best parameter settings and strategy for a dataset and a cost matrix individually can improve upon results than when using the same parameter settings for every cost matrix allocated to a dataset. Finally Section 5.4 attempts to find guidelines which will determine the best combination of parameter settings for a dataset and its allocated cost.

5.1 Parameters allowing continuation of process when it is worthwhile

The table in Appendix Section A5 gathers statistics for each of the parameter settings to determine which ones produce highest frequency of best and worst results. Out of all the parameter settings, the pre-pruning options which carry out tests in order to see whether it is worthwhile carrying on with the process have predictably the greater control when it comes

to decreasing costs and increasing the accuracy rates. However the control for stopping early when the information in the dataset is reduced i.e. proportional stopping of both the examples and dictating the class proportion does not have as much impact as the other pre-pruning option. This is interesting as it was expected that proportional stopping would have more impact than it does.

Lower costs are most often produced by the proportional stopping option than by allowing full tree growth but this includes the pre-pruning combinations. The four pre-pruning combinations are; a: true, true; b: true, false; c: false, true and d: false, false; where the first Boolean value dictates whether the method described by Ling et al. (2004) is carried out and the second Boolean value dictates whether the attribute chosen is tested to see if it is worthwhile or not. The combination 'c' produces the majority of better low cost results, followed by 'b'. Each of these only carry out one test; the combination 'c' does not carry out Ling et al. (2004)'s method, only the best attribute is tested whilst the other combination 'b' is the reverse. For accuracy, no proportional stopping along with the combination 'd' produces the majority of the better results. It is thought that trees are quite often not built when the pre-pruning combination 'a', 'b' and 'c' all return the same low cost and low accuracy. Evidence suggests that the pre-pruning combination options control tree build rather than the proportional stopping option as there are many results which are the same regardless of whether the proportional stopping option has been used. The pre-pruning combination option 'd' allows the proportional stopping options to help raise accuracy and/or lower costs. For example the accuracy is raised when the pre-pruning combination 'd' is used, but with the addition of proportional stopping, the accuracy is slightly reduced and with a lower cost. By examining the table in Appendix Section A5, it is clear that the pre-pruning combination has most effect on whether a tree is grown or not. The combinations a, b or c are

checking whether it is worthwhile or not. The pre-pruning combination option 'a' produces less trees than the other combinations which is to be expected as it performs more checks. The pre-pruning combination option 'd' is where the algorithm is forced to grown a tree. For some datasets, namely breast, diabetes and hepatitis, employing the proportional stopping option increases the accuracy rates. When these datasets are allowed to be processed fully, the extra information used does not help to improve upon the accuracy rates unlike the other datasets.

As mentioned earlier, when the algorithm considered it not worthwhile to continue, it can sometimes result in no model being built. In order to investigate how building a model can improve results, a series of graphs have been compiled which (i) plot the 'do nothing' cost from all examples and those from where examples have a model, (ii) plot 'do nothing' accuracy from all examples and those from where examples have a model, (iii) plot the mean cost obtained from all examples and from where examples have a model (iv) plot the mean accuracy obtained from all examples and from where examples have a model.

Figure 17 presents the graphs for costs whilst the accuracy graphs are in the Appendix D1. A black solid line indicates do nothing mean cost for all examples, a black dotted line indicates the do nothing mean cost for only those examples producing a tree, a green solid line indicates the mean cost value obtained from all examples whilst the red dotted line the mean cost value obtained for only those examples producing a tree. The same style of lines is used for the corresponding accuracy values.

It is hoped that by using different combinations of parameter settings, a combination of them can be found that will increase accuracy and lower cost at the same time. The relationship

between test costs and misclassification costs are significant. Based on earlier observations, it can be expected that if the test costs are lower than the misclassification costs, it makes sense to grow a model in order to reduce these higher costs. If the reverse is the case it is only worthwhile growing a tree if the misclassification costs were to be greatly reduced but not at the expense of spending test costs. When the test costs and misclassification costs are a mixture of high and low amounts, it is harder to say whether trees should be grown or not, it is likely that smaller trees would be required and this would need exploration to find the correct combination of attributes in the tree.

In general the red line is always a higher cost except in the 2-class graphs, where the cost returned from the tree examples only is lower than the cost returned from all examples for some cost matrices, those where the misclassification costs are low. In the 3-class graph, for the misclassification costs from the group which is higher than the test costs; 7, 8, 9 the cost returned from the tree examples only is lower also. This is repeated for the 5-class graph for costs for some cost matrices. For the 15 class graph, there is no red line as all examples where produced using trees so it would be identical. In this case, growing a tree proved better for all cost matrices than doing nothing.

As was suggested, when the misclassification costs are much lower than the test costs, there is a reluctance on the part of the algorithm to produce a tree, this is evident in that the values for the costs obtained (green line) are very close to the ‘do nothing’ costs for the misclassification costs where this is the case.

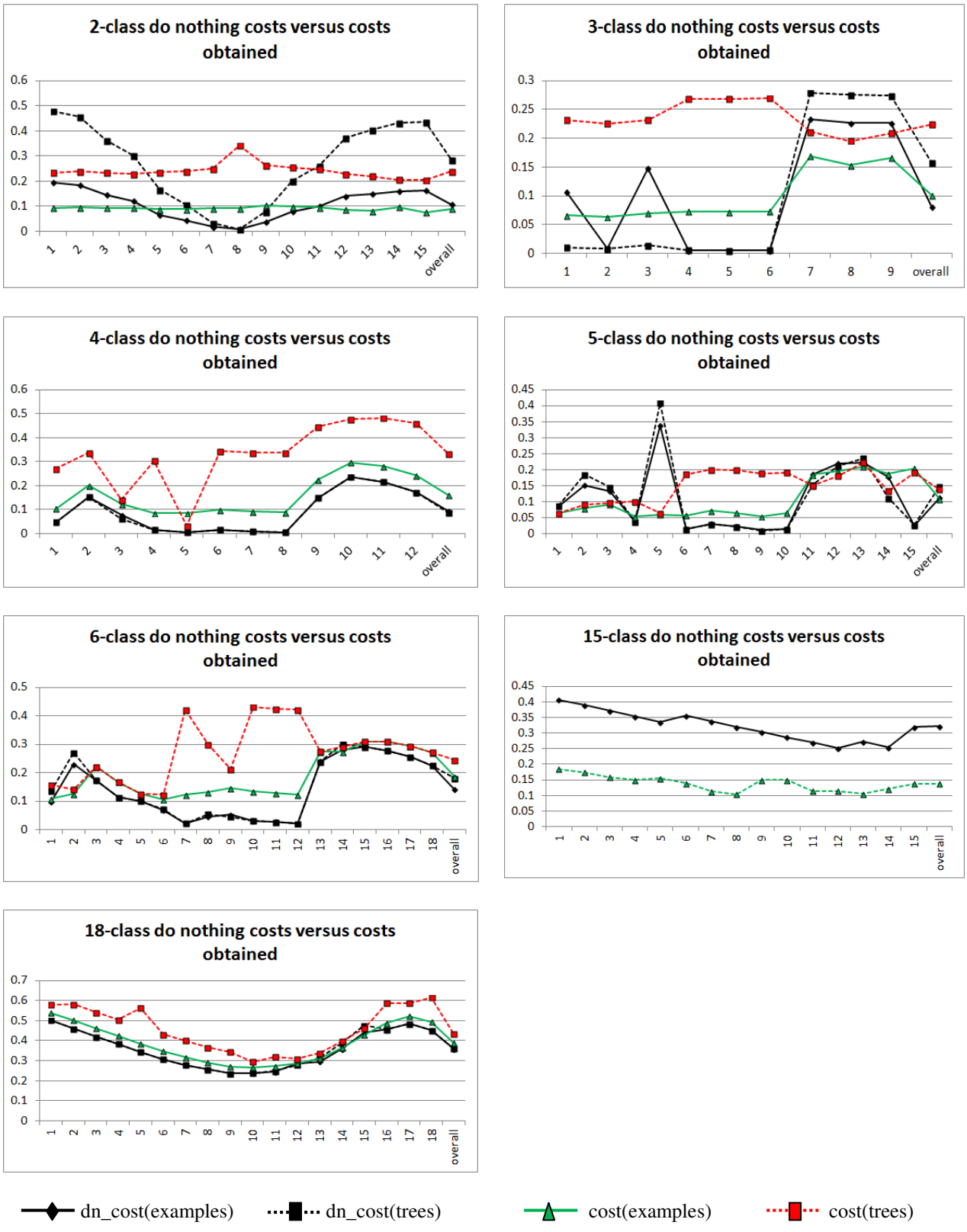


Figure 17 Graphs showing do nothing costs versus cost obtained for each of the types of classes

The interaction between the higher and lower misclassification and test costs is easier to see when the misclassification costs are averaged over their 3 groups; the binary classes datasets have to have their misclassification costs averaged over the whole of the matrices. Graphs have been compiled for each of the multi-class datasets for the same series of values but the misclassification costs have been grouped into their types; mixed, low and high in relation to the test costs.

Most of the datasets follow the pattern where the costs obtained are higher than the 'do nothing' costs and the accuracy obtained is higher than the 'do nothing' accuracy. Figure 18 presents the exceptions to this, which are the annealing, glass, iris and wine datasets for the group of misclassification costs which are higher than the test costs. The annealing and nursery datasets show the same pattern for the group of misclassification costs which have values both higher and lower than the test costs. The remainder of the graphs are presented in the Appendix D2.

For those datasets where the misclassification costs can only be averaged to one group which is a group with mixed high and low costs in relation to the test costs, graphs with the values of 'do nothing' values and values obtained are also presented in the Appendix D2. All the krk results are high in relation to the other costs and the mushroom results are low, and their accuracy rates are low and high respectively. The cost value for the breast dataset is lower than the 'do nothing' cost value however the corresponding accuracy returned is lower than the 'do nothing' accuracy, which is unusual.

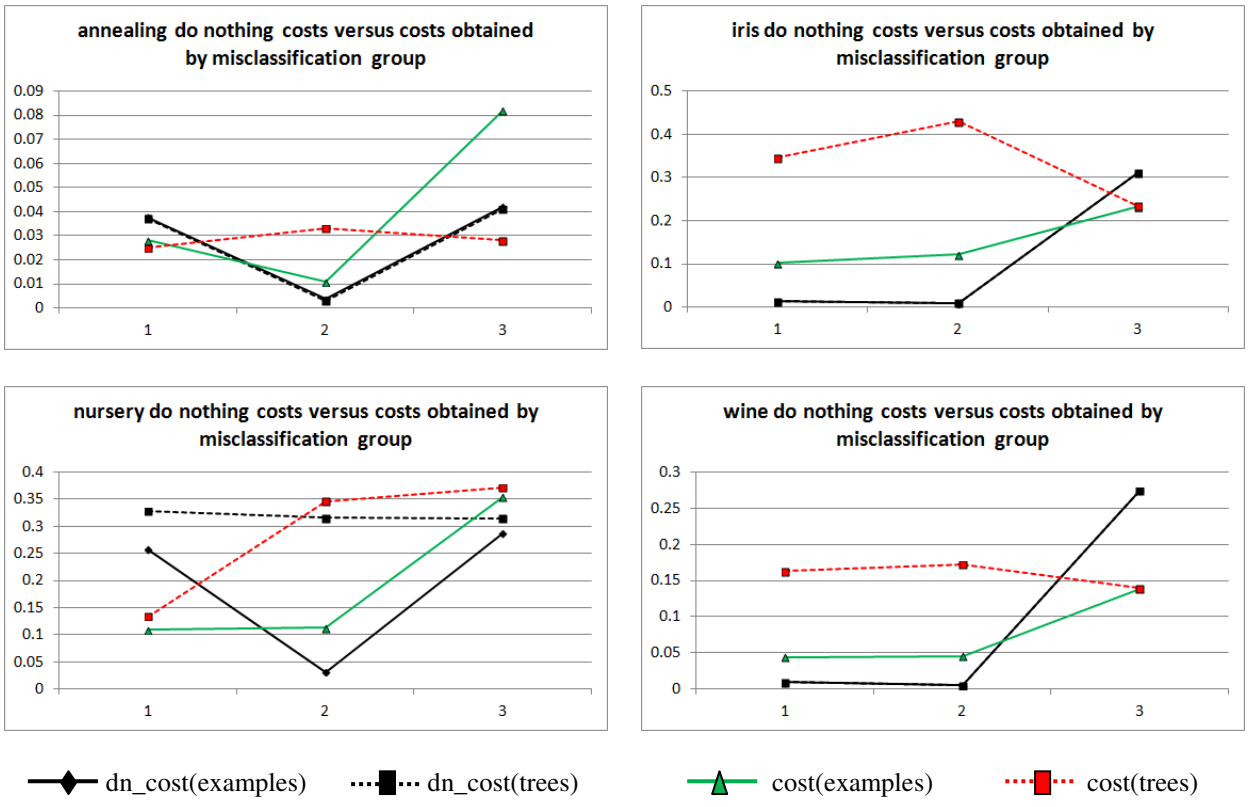


Figure 18 Graphs showing multi-class datasets in their 3 groups of misclassification costs: mixed, low, high

To obtain higher accuracy rates, it is necessary to grow a tree. Although this more often than not results in a higher cost, the aim of the algorithm is to minimize sacrificing the accuracy rate. In order to do so a tree must be grown. The more exploration that takes place, the more likely it is to produce a model which will meet this aim and still reduce costs. One must remember that in order to classify something it will be necessary to pay some cost. Being able to alter the algorithm's behaviour is necessary to achieve this aim. It is acknowledged though, that the needs of the user will depend on the domain, and not producing a model is an indication of the fact that it may not be worthwhile to do so. This is vital information so that costs are not wasted, and other techniques can be used instead.

5.2 Determining how many lever pulls, which version and strategy is desirable

The multiple-model version of the algorithm using the cost-based strategy produces the majority of the best results over all datasets. The preferred value of P lever pulls is either the same as or greater than the potential unique bandit path value as a general rule.

Although the identifier of P lever pulls refers to values which are personal to each dataset, they are all in relation to the potential unique bandit paths of the dataset. So for example PId 3 is the potential unique bandit path value, PId 4 is a value which is higher than the potential unique bandit path. Six of the datasets need a higher amount than the potential unique bandit paths to produce good results, these are breast, car, diabetes, glass, krk and soybean. The others can produce good results with a lower value. For mushroom it does not matter, although all values of P are very high anyway for this dataset. Investigation with regard to the depth to look-ahead, of the four datasets chosen to look to a depth of 2, although good results were obtained for the iris and diabetes datasets, these results were also obtained using only a depth of 1, the glass dataset showed no improvement when looking to a lower depth and there was no improvement of results for the car dataset either. The conclusion is that as these datasets required only a relatively small value of P as their potential unique bandit path values, perhaps the datasets did not benefit from looking ahead to a lower depth. It is possible however that datasets with a larger value of P as their potential unique bandit path values might benefit from looking ahead to a lower depth.

With regard to the attribute with one value strategies, the ignoring or find next best options are those which are in the majority for producing good results; however it only makes a difference if the criterion is actually met. It is possible that this criterion is seldom invoked for the majority of datasets. Perhaps it is better to examine the dataset to determine whether a

dataset is likely to produce this scenario and then remove attributes which are similar in many ways which duplicate information and do not add anything new.

There is a higher frequency of Class 1 examples using the hybrid labelling system than using the cost-sensitive labelling system. Using the cost-sensitive labelling always reduces the accuracy in the unbalanced datasets; the more balanced the dataset, the less the sacrifice against the two labelling systems because of course, when the class is balanced, the cost-sensitive side of the hybrid labelling system will be used anyway.

5.3 Investigate taking advantage of different parameter settings to achieve aim

It has been observed that some parameter settings produce good results for some cost matrices, but the same settings do not produce as good results for others. Section 4.3 has discussed the concept of treating each cost matrix for each dataset as an individual entity and to find the most suitable settings producing the best results for it.

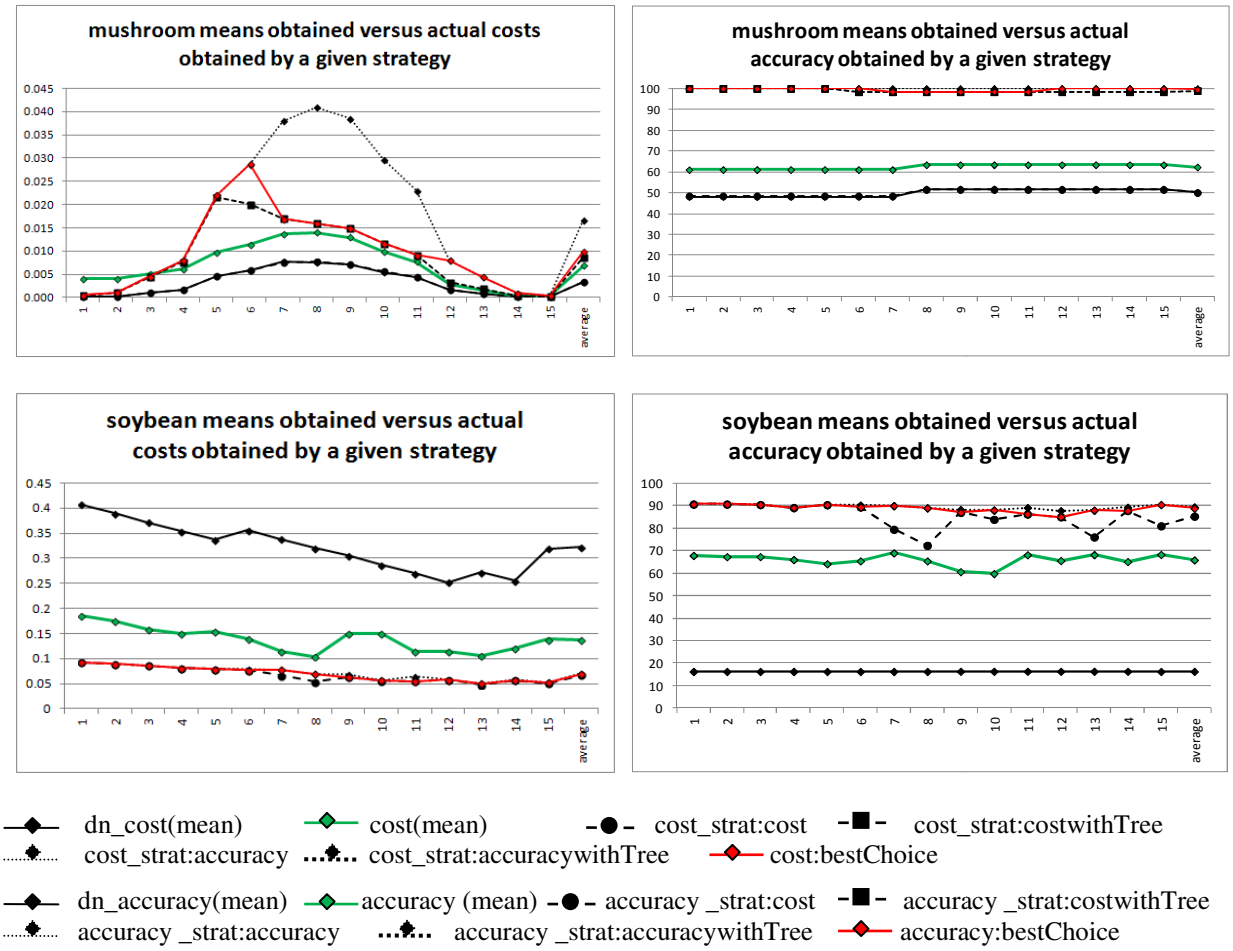
It is possible, therefore, to improve on the results of a dataset by considering separately the best parameter settings. So for example instead of thinking about a dataset and finding parameters for it, it is much better to think of it as different datasets, one for each of the cost matrices it has been allocated. For example, the annealing dataset is one dataset, with 15 cost matrices. Better results can be obtained if it is thought of as 15 datasets. Therefore in these experiments 210 datasets are processed each as individual entities, each with their own best combination of parameter settings in order to achieve the best results possible.

In addition to considering parameter setting combinations, the strategy used to choose the parameters needs to be considered. Should the parameter settings which produce the lowest cost be chosen and then use the corresponding accuracy, or should the parameter settings which produce the highest accuracy be chosen and then use the corresponding cost. Maybe, depending on the domain and needs of the user and the relationship between test costs and misclassification costs, a mixture of these should be chosen in order to try to achieve the aim of cost-sensitive learning; that of maximizing accuracy at the lowest cost possible. A full set of tables which are the best values achieved for a cost matrix for a dataset are given in the Appendix D3. For each dataset and cost matrix, the values obtained by each experiment were examined. The example with the lowest cost is selected, and the example with the highest accuracy is also selected along with their corresponding accuracy/cost. If either of these examples did not produce a tree, then an alternative example is also chosen which has the next lowest cost obtained from producing a tree or the next highest accuracy producing a tree along with corresponding values. Examples which are chosen by both strategies are indicated in the tables. The parameter settings used to produce each of these results are also detailed in the tables.

Figure 19 illustrates that different cost matrices may require different strategies. Mean values have been calculated for scenarios ‘do nothing’ that is if no model has been induced and the overall value obtained if no strategy is chosen but simply the mean value obtained from all examples. Additionally the values obtained when using the cost-based strategy and using the accuracy-based strategy are plotted.

In order to attempt to anticipate what a user might require, ‘best choice’ illustrates the differing needs of the cost matrices; the costs and accuracy rates from both strategies are

examined, trying to allow for user requirements and to achieve low costs whilst minimizing sacrificing the accuracy rate.



For example, cost_strat:cost is the lowest cost obtained and cost_strat:accuracy is the corresponding accuracy; accuracy_strat:accuracy is the highest accuracy obtained and accuracy_strat:cost is the corresponding cost; accuracy:bestChoice is the accuracy rate from the chosen strategy and cost:bestChoice is the corresponding cost

Figure 19 Comparing mean values obtained with values obtained by a given strategy

The solid black line represents those values which are the result of doing nothing, that is inducing no tree at all and the green line represents the mean values which have been calculated using all of the examples in the results dataset and the red line represent the values representing 'best choice'. The accuracy rate, which corresponds to the choice of the cost-based strategy, is examined with the accuracy rate obtained from the accuracy-based strategy. If the difference between these is greater than 3%, the sacrifice is considered too great for a

cost matrix so the accuracy-based strategy value is chosen for that cost matrix and the corresponding cost used. In order to achieve lower costs whilst maintaining accuracy or minimizing sacrifice, a compromise is sought between the two strategies, if this is difficult to meet, the flexibility of the algorithm means that users have more choice than with traditional cost-sensitive algorithms. It may be that the user wants to go for a more cautious approach to cost expenditure and err on the side of caution.

The best illustration of the idea behind choosing the appropriate parameter settings for each individual run of a dataset as described are the mushroom and soybean dataset in Figure 19. First the mushroom dataset, which maintains an accuracy rate of 100% over the majority of the cost matrices and achieves this at a lower cost. For cost matrices 7 to 11, the cost-based strategy with tree and accuracy-based strategy both produce very high accuracy rates, but the cost-based strategy returns these at a lower cost. As the difference between the accuracy rates for these two strategies is only 1.48% but the cost is less by 0.014 to 0.021 the strategy for the 'best' choice for these cost matrices is the lowest cost and corresponding accuracy; the cost-based strategy.

For the soybean dataset the pattern is the ideal pattern; if nothing is done i.e. no model induced, cost is high because the accuracy is low and there are misclassification costs. When a model is built, the mean cost value returned for each of the cost matrices is much lower and the corresponding accuracy is higher. When values are chosen which meet the cost-based strategy, the cost is lower and the accuracy increases. When values are chosen which meet the accuracy-based strategy, the accuracy is high and the cost is even lower. Appendix Section D4 presents the rest of the results for each dataset.

Examining the patterns for the remaining datasets, for the car, glass, heart, iris and krk datasets, the red line showing the 'best' choice is higher for costs than the other options, but the accuracy is improved. The annealing, breast, diabetes, flare, hepatitis, nursery, tic-tac-toe and wine show that the 'best' choice can reduce costs and still keep the accuracy high for at least some cost matrices.

5.4 Developing guidelines for datasets to determine the best combinations of parameter settings

As there are a large number of potential combinations of parameter settings which could be used when processing a dataset, it would be useful to see if the best settings for a dataset could be predicted as a guideline for processing future datasets. In order to do this, two approaches have been used. The first way looks at the parameter settings to see which of their values resulted in the highest frequency of the Class 1 examples. The interest is only in Class 1 examples as these are the best results and it is only these for which it is important to predict the settings required to achieve them. There must be some way to describe the dataset and the costs in order to determine the parameter settings which would be best. Two attributes in the results dataset which can help with this are 'grpPotPaths', which groups together potential unique bandit path values and 'grpRatio' which groups the relationship between the highest misclassification cost and the highest test cost used by an experiment. By examining each of the attribute values and examining which parameter settings lead to the highest frequency of Class 1 results, the following information has been compiled in Table 12.

description of dataset	strategy	no of models	label system	depth (*k*)	one value strategy	id of P value	pre prune option	prop stopping	class prop stopping
if potPaths <= 500	accuracy	10	HYBRID	1	2	4	c	TRUE	TRUE
if potPaths <= 1000	cost	10	HYBRID	2	2	4	d	FALSE	FALSE
if potPaths <= 2000	none (0)	1	HYBRID	1	0	5	d	TRUE	TRUE
if potPaths <= 3000	none	1	HYBRID (COST)	1	2	3/4	a (d)	TRUE	TRUE
if potPath <= 5500	cost	10	HYBRID	2	2	2/3	a (d)	TRUE	TRUE
if potPath <= 9200	cost	10	HYBRID	1	2	3	d	FALSE	FALSE
if potPath = all other values	any	1 or 10	either	1	any	any	d	either	either
grpRatio <= 0.1	none	1	HYBRID	1	2	5	d	FALSE	FALSE
grpRatio <= 0.99	none	1	HYBRID	1	2	5	d	TRUE	TRUE
grpRatio is 1.0	accuracy	20	LABEL	1	2	4	d	TRUE	TRUE
grpRatio <= 10.0	cost (none)	10 (1)	HYBRID (COST)	1	2	5	b (c)	TRUE	TRUE
grpRatio < = 100.0	accuracy	20	HYBRID	1	2	4	d	FALSE	FALSE
grpRatio <= 500.0	accuracy	20	HYBRID	1 (2)	0	4	d	FALSE	FALSE
grpRatio <= 1000.0	cost	10	HYBRID (COST)	1	0	2	d	FALSE	FALSE
grpRatio > 1000	cost	10	HYBRID (COST)	1	0	2	d	FALSE	FALSE

Table 12 Manually obtained parameter settings based on description of a dataset and frequency of class 1 examples

The second way is to use a technique known as ‘meta-mining’. Attributes representing the parameter values, class distribution, grouped potential unique bandit paths and the ratios of misclassification cost to test cost along with the class to which the example has been assigned were selected and following the experimental methodology described earlier in Section 4.1, were processed by the accuracy-based algorithm J48 in WEKA.

The WEKA J48 was run on three kinds of files; (i) all examples, (ii) examples from multiple-model version, (iii) examples from single-model version. Each kind of file was executed with different pruning levels in order to produce a more compact tree but one with good accuracy. The confidence levels entered were the default value, 0.15 and 0.05. The accuracy rates obtained are presented in Table 13.

Out of the 9 options c0.05 - all examples has the highest accuracy of 82.95%. Using the most extreme version of pruning produced the highest accuracy for each type of file. The lowest accuracy returned was from the single version examples using the default pruning level.

J48 with confidence level	accuracy obtained
c0.05 – all examples	**82.95423**
c0.05 – ensemble	82.92961
c0.05 - single	82.45773
c0.15 – all examples	82.89284
c0.15 – ensemble	82.79647
c0.15 - single	82.38293
default – all examples	82.53402
default – ensemble	82.43935
default - single	82.16518

Table 13 Accuracy rates obtained from J48 predicting classes from parameter settings

No matter what file is examined; the root attribute is always 'distribution' which describes the class distribution of the dataset. The ten files producing the better results have been averaged by class i.e. each individual class has had its accuracy rate calculated. Out of all Class 1 examples, 91% of them were predicted correctly from their parameter settings using descriptions of the dataset. Of the ten files, the one which contains the highest accuracy has been used to extract rules to decide on parameter settings as guidelines. This file contains the highest overall accuracy of 83.12% and with an accuracy of 94.3% on predicting Class 1 examples. Leaves not being labelled Class 1, i.e. good cost, good accuracy, were removed leaving around 106 leaves from 557 leaves. If attribute 'tree' has been chosen for a node, the

branch indicating no tree was grown has been ignored in favour of following the path where a tree has been grown as it is necessary to concentrate on parameter settings where a tree has been grown. Even with harsh pruning there are still leaves with few examples. Any leaf, having only 100 or less examples, is also removed, leaving the best and most confident leaves to be used. Leaves with a high percent of accuracy are kept; in order that all root node branches have relevant leaves, this has been set at 65%. There are 33 leaves remaining and these are used to extract rules. The rules are shown in Figure 20.

When comparing these two approaches it is observed that there are differences between them. Although both attributes in the manual approach are used in the 'meta-mining' approach, two of the attributes chosen to help determine parameter settings have one thing in common. They both describe the nature or structure of the dataset. The attribute 'grpPotPaths' has grouped potential unique bandit paths in value ranges. Potential unique bandit paths use attributes and their values in the calculation so could be a good indication of how attributes and their different values interact. The attribute 'balanced' describes the class distribution so between these two attributes the dataset's structure is described. The other attribute, which was the ratio between misclassification costs and test costs, describes the other important aspect with regard to the dataset; the costs which are to be used.

By using the results dataset, the rules obtained from WEKA as described and the information manually obtained, have been examined. The rules from WEKA prove the most accurate. However as this produces combinations of parameter settings and the manual version does not, this could be anticipated. The manual selection of ways to describe datasets should not be dismissed so easily however, as it could be used in combination with the other technique.

IF distribution is even
IF multi-class is FALSE
IF grpPotPaths is >500 & <= 1000
Set –prop FALSE
Set –label COST
OTHERWISE
Set –prop TRUE
Set –label HYBRID
IF multi-class is TRUE
IF grpPotPaths <= 500
IF ratio >1.4
Set –pre-pruning combination a
OTHERWISE
Set –pre-pruning combination d
Set –prop TRUE
IF grpPotPaths >500 & <= 1000
IF ratio < 6.02
Set –pre-pruning combination a
Set –depth 2
OTHERWISE
Set –pre-pruning combination d
Set –depth 1
IF grpPotPaths >1000 & <= 2000
Set –pre-pruning combination b
Set –model 10
OTHERWISE
Set –depth 1
Set –pre-pruning combination d
Set –prop FALSE

IF distribution is part unbalanced
IF ratio > 7.14
Set –label HYBRID
Set –prop FALSE
Set –pre-pruning combination d
OTHERWISE
Set –label COST
Set –prop TRUE
Set –pre-pruning combination d

If a value has not been found for a parameter, this is an indication that it is not statistically important so therefore it is suggested that the default value be used.

IF distribution is 65/35
IF ratio > 7.0 & <= 71.0
Set –strat accuracy
Set –pre-pruning combination b
IF ratio > 71.0 & <= 219.0
Set –prop TRUE
Set –pre-pruning combination d
Set –label HYBRID
IF ratio > 219.0
Set –prop FALSE
Set –pre-pruning combination d
Set –label HYBRID
OTHERWISE
Set –strat cost

IF distribution is almost even
IF grpPotPaths is >= 6000 & <=9200
Set –strat accuracy
OTHERWISE
Set –strat cost

IF distribution is unbalanced
IF grpPotPaths <= 500
Set –label HYBRID
Set –prop TRUE
IF grpPotPaths >500 & <= 1000
Set –oneVal stop
Set –model 10
Set –prop FALSE
IF grpPotPaths >1000 & <= 2000
IF ratio <= 21.94
Set –oneVal stop
Set –prop TRUE
IF grpPotPaths > 3000 & <= 5500
Set –label HYBRID
Set –pre-pruning combination d
Set –prop TRUE
OTHERWISE
Set –label COST
Set –oneVal nextBest
Set –prop TRUE

Figure 20 Rules extracted using J48 accuracy-based algorithm on examples in the results analysis file

The guidelines produced may prove useful to determine how to set the parameter values. Any of the three identified attributes could be used to obtain the parameter values but it would be important to conduct some experiments in order to see if they are as successful as was indicated.

So for example an unseen dataset can be chosen, test costs and misclassification costs assigned and potential unique bandit paths calculated. Then each of the three attributes could be used in turn to set parameter settings and then execute the appropriate algorithm using the parameter settings recommended. Again all possible parameter settings in combination can also be executed and results compared. This would be a good test of whether guidelines can indeed be recommended.

5.5 Summary of findings from the investigation

An extensive investigation, carried out on the 15 real-world datasets, resulted in 158664 experiment results which have been collated into a results dataset and analyzed using a statistical software package (SPSS). Each experiment has been carried out used different strategies using all possible combination of parameter settings. Each example in the results dataset has been allocated a class based on the combination of the cost and accuracy rate returned by that particular example. Both versions of the algorithm were used; the single version producing one model and the multiple-model version producing *n* models.

The aim has been to determine which parameters allow the algorithm to only continue when it is worthwhile to do so and the effect this has on cost and accuracy, to determine how many lever pulls should be used for each dataset, and which version of the algorithm and which

strategy works best. Another aim is to investigate how to take advantage of the different parameter settings and strategies and to see if guidelines for the setting of these parameters can be found.

Sometimes, when the algorithm considered it not worthwhile to continue, resulted in no model being built. This led to an investigation into the scenario of a model or no model grown. Graphs show that accuracy rate is improved when a model is generated. Obviously costs tend to be higher when a model is grown but in some cases, particularly in multi-class datasets like soybean, growing a model helps to reduce costs, when the misclassification costs are greater than the test costs.

Two techniques were explored to see which may be able to help find guidelines to parameter settings for unseen datasets. Using the class representing the best results, parameter settings have been examined to see which ones produced the highest frequency in order to determine whether these parameter settings are best for a dataset, which has a certain number of potential unique bandit paths or has a particular ratio of misclassification costs and test costs.

The other technique used, called 'meta-mining', finds rules using the accuracy-based algorithm J48, which could be used in order to find guidelines to parameter settings. Each used an attribute which was an indication of the structure of the dataset, for example, class balance and potential unique bandit path value and additionally the 'meta-mining' technique used an attribute which uses the ratio of the misclassification costs and test costs. It is thought that a combination of the two techniques would be helpful.

From the investigation of strategy and parameter settings choice, if each dataset/test cost/misclassification cost combination is treated as a separate dataset and the best parameter settings and strategy for each one are found, this makes the algorithm under development very flexible in comparison with existing cost-sensitive algorithms because it can behave differently when given different parameter settings. This flexibility can be used to improve results.

From each experiment, the lowest cost along with its settings for each cost matrix has been examined to find the best one overall (cost-based strategy) and the same for the highest accuracy (accuracy-based strategy). These values are carried forward into an evaluation of existing algorithms. It is necessary to check the development using these 15 real-world datasets and compare performance measures against existing well-known cost-sensitive algorithms.

CHAPTER 6: AN EMPIRICAL COMPARISON OF THE NEW ALGORITHM WITH EXISTING COST-SENSITIVE DECISION TREE ALGORITHMS

In order to examine the Multi-Armed Cost-Sensitive Decision Tree algorithm (MA_CSDT) and compare performance with well-known existing cost-sensitive algorithms, the 15 datasets along with the test costs and misclassification costs used in the investigation (see Chapter 5) are used in a comparison along with J48 (C4.5 version 8) in WEKA in order to see if the aim of the algorithm can be achieved over the wide variety of datasets and costs.

The algorithms chosen for comparison, along with J48 are *EG2* (Núnez 1991), *MetaCost* (Domingos 1999), *AdaCostM1* (Fan et al. 1999), *ICET* (Turney 1995) and *ACT* (Esmeir and Markovitch 2008). The algorithm J48 is chosen to provide a benchmark for accuracy and the cost-sensitive algorithms selected represent five classes of cost-sensitive decision tree algorithms as described in Chapter 3 and in Lomax and Vadera (2013). Four of the algorithms have been implemented in the open source data mining software package WEKA (Hall et al. 2009) and they have been adapted to include test costs in their evaluations.

- *EG2*, described in Section 3.1.1.1 has been implemented using the description given in Turney (1995) in which the J48 implementation has been adapted to include equation (3.1).
- *MetaCost* is an algorithm which is included in the WEKA package, described earlier in Section 3.2.2.2.
- *AdaCostM1* is an adaptation of the algorithm AdaBoostM1 (Freund and Schapire 1996) which is included in the WEKA package. The adaptations developed by Fan et al. (1999) for the algorithm *AdaCost*, and described in Section 3.2.2.1 have been

added to AdaBoostM1 in order that the algorithm *AdaCost* can process multi-class datasets and be included in the evaluation.

- *ICET*, as described in Section 3.2.1 has been previously implemented and has been tested by comparing experiments in Turney (1995) in order to check the implementation (Vadera and Ventura 2001). This implementation has been used in several previous studies.
- *ACT*, as described in Section 3.2.3 has been implemented using the description given in Esmeir and Markovitch (2008) and has been tested by comparing experimental results detailed in their paper.

The J48 algorithm produces the target accuracy rates, in order to determine whether a compromise between accuracy-based decisions and cost-based decisions can be found so that it is possible to induce decision trees that maintain accuracy but also minimize costs. The results obtained from the experiments investigated in Chapter 5 have been examined in order to determine whether the hypothesis can be confirmed. As there are two different approaches to take while examining the results, both of these approaches have been used to produce values to compare. Over all the examples in the results dataset, lowest cost for a dataset and cost matrix is chosen along with its corresponding accuracy rate and is designated as COST-BASED(ALL). Additionally the highest accuracy along with its corresponding cost is also chosen and is designated as ACCURACY-BASED(ALL). This is repeated using only those examples where the proportional stopping parameters are set to TRUE and using only those examples where the parameters are set to FALSE. These are labelled COST-BASED<> and ACCURACY-BASED<>. The MIXMATCH strategy examines the described strategies and picks one for an individual cost matrix which may help improve upon the results obtained by each of the cost-sensitive algorithms. For example, cost matrix 1 for a dataset, uses COST-

BASED<> as it is a better choice. For cost matrix 5 for the same dataset, ACCURACY-BASED<> might be the better choice rather than staying with COST-BASED<> for all of the cost matrices. The MIXMATCH strategy attempts to combine the strategies in a flexible way in order to achieve the aim of the algorithm.

To examine cost and accuracy from the five cost-sensitive algorithms, processing the datasets using both pruned and un-pruned versions of *EG2, MetaCost, AdaCostM1* and *ACT* have been used with *ICET* using only a pruned version, as the un-pruned version never produces better results than the pruned version.

As the MA_CSDT algorithm, using some combinations of parameter settings, resulted in no trees being induced, in the first instance the output of the six algorithms has been examined to see whether trees have been produced. It is likely that, in some cases either pruning the tree results in no tree being left or that no tree has been grown in the first instance. Table 14 presents the percent that trees were not grown during the processing of the datasets using pruned versions of the algorithms. Additionally it gives the misclassification cost identifiers which have produced the least trees. For example, in the breast dataset, all of the misclassification cost matrices processed by the *EG2* algorithm, had an equal rate of not producing trees. For the *ICET* algorithm, the cost matrices 1 to 12 did not produce trees. The algorithms *MetaCost* and *AdaCostM1* only produce trees for the misclassification cost identifier 8. Comparing these to processing of the datasets using un-pruned versions, there are still some algorithms for some of the cost matrices which do not produce a tree. These are *MetaCost* and *AdaCostM1*. It is concluded that the other algorithms did not produce a tree owing to pruning reasons, however for *MetaCost* and *AdaCostM1* the reason that no trees have been produced on some occasions, is that the process stops as all the examples belong to

Dataset (total no of times files processed)	J48	EG2	MetaCost	AdaCostM1	ICET	ACT	Total % trees not grown
annealing (150)							
% tree not grown	0	0	0	0	0	33.33%	5.55%
CM: least trees	n/a	n/a	n/a	n/a	n/a	6-10	
breast (150)							
% tree not grown	0	30.00%	93.33%	93.33%	78.66%	20.00%	52.55%
CM: least trees	n/a	all equal	not 8	not 8	1-12	7-9	
car (120)							
% tree not grown	0	0	0	10	87.50%	58.33%	24.31%
CM: least trees	n/a	n/a	n/a	4	1,3,5-8, 11	5-8,10-12	
diabetes (150)							
% tree not grown	0	0	86.66%	86.66%	18.66%	6.66%	33.11%
CM: least trees	n/a	n/a	1-6, 9-15	1-6,9-15	4,6-8	8	
flare (90)							
% tree not grown	70.00%	100.00%	63.33%	52.22%	84.44%	68.89%	73.15%
CM: least trees	all equal	all equal	2	2,5	4,7,8	4-9	
glass (180)							
% tree not grown	0	0	0	0	1.11%	60.56%	10.28%
CM: least trees	n/a	n/a	n/a	n/a	5,11	8 -11 ,15-18	
heart (150)							
% tree not grown	0	0	80.00%	82.67%	0	6.67%	28.22%
CM: least trees	n/a	n/a	1-5,10-15	1-6,10-15	n/a	8	
hepatitis (150)							
% tree not grown	0	0	80.00%	82.00%	3.33%	6.67%	28.67%
CM: least trees	n/a	n/a	1-5,11-15	1-7,12-15		8	
iris (90)							
% tree not grown	0	0	0	0	14.44%	35.56%	8.33%
CM: least trees	n/a	n/a	n/a	n/a	1-6	3-6	
krk (180)							
% tree not grown	0	0	0	0	2.22%	22.22%	4.07%
CM: least trees	n/a	n/a	n/a	n/a	15	1,3-5,16-18	
mushroom (150)							
% tree not grown	0	0	0	40.00%	0	6.67%	7.78%
CM: least trees	n/a	n/a	n/a	1-4,14-15	n/a	8	
nursery (150)							
% tree not grown	0	0	0	0	15.33%	40.67%	9.33%
CM: least trees	n/a	n/a	n/a	n/a	1,2	8-11,13	
soybean (150)							
% tree not grown	0	0	0	0	0	0	0
CM: least trees	n/a	n/a	n/a	n/a	n/a	n/a	
tictactoe (150)							
% tree not grown	0	0	32.67%	86.67%	22.00%	6.67%	24.67%
CM: least trees	n/a	n/a	1-4	1-7,10-15	13	8	
wine (90)							
% tree not grown	0	0	0	0	0	33.33%	5.56%
CM: least trees	n/a	n/a	n/a	n/a	n/a	4-6	

Table 14 Percent of trees not grown by dataset for the six algorithms in the evaluation with the cost matrices producing the least trees

one class. In the case of *AdaCostM1* this will be owing to the initial weight procedure and in the case of *MetaCost*, owing to the way that this algorithm re-labels the training example with the class that minimizes the cost, as described in Section 3.2.2.2.

The *ACT* algorithm is the only one of these six which fails to grow trees for the majority of the datasets. In fact the only dataset where this algorithm grew a tree for each training and testing pair over each cost matrix, is the soybean dataset. For this particular algorithm the most likely explanation of this is owing to its strict pruning policy. The other algorithms use error-based pruning, which involves testing whether having a sub-tree results in more errors than if it were to be pruned to a leaf node. The *ACT* algorithm uses an extension of this method to include the misclassification costs and the test costs. Only if the cost is reduced by having a sub-tree, is the sub-tree retained. Therefore, in some cases the *ACT* algorithm considers it not worthwhile to retain the sub-trees, reducing the model to the original subset. Two-class datasets and 3-class datasets have the fewest trees grown over all the processed files. The fact that trees are not always grown will be taken into account when looking at the costs in the evaluation. Section 6.1 presents the results of the evaluation and Section 6.2 presents a discussion of the outcome of the evaluation.

6.1 Empirical comparison results

Table 15 presents the percentage that each cost-sensitive algorithm achieved the lowest cost for a cost matrix or the highest accuracy for a cost matrix for each of the datasets examining both pruned and un-pruned versions of the algorithms.

	lowest cost for a cost matrix	highest accuracy for a cost matrix
J48	1.38%	32.08%
EG2	9.67%	11.77%
MetaCost	6.45%	11.09%
AdaCost	3.92%	3.92%
ICET	5.07%	3.41%
ACT	5.30%	2.73%
MA_CSDT	68.20%	34.98%

Table 15 Percent that each cost-sensitive algorithm achieves the lowest cost or highest accuracy for a cost matrix for both pruned and un-pruned versions

An algorithm returning the highest accuracy must have a lower cost than the J48 corresponding cost. If more than one algorithm qualifies for either lowest cost or highest accuracy, each one is included in the percentage shown.

Table 16 presents a summary of each dataset showing whether the MA_CSDT algorithm has met its main aims. The main aims are (i) the MA_CSDT returned the same accuracy rate as J48 or higher; (ii) in order to obtain the same or higher accuracy rate, a lower cost has also been returned; and (iii) produces the lowest cost than all other algorithms. For the majority of the datasets these aims have been met. There are a number of datasets where a simple strategy of choosing the highest accuracy and corresponding cost for all cost matrices produces good results which meet the aims of the algorithms by returning a higher accuracy rate than J48 at a lower cost, and a number of datasets where choosing a mixture of different strategies for the cost matrices produce good results. The exceptions are the car, krk and nursery datasets where no strategy of MA_CSDT has been able to produce an accuracy rate comparable with the accuracy-based algorithm J48.

	accuracy same or higher than J48	if no % sacrifice	is this cost lower than J48	produces a lower cost overall	if yes % sacrifice	comments regarding the evaluation
annealing	**can combine strategies to improve results*					
pruned*	yes		yes	no		*EG2 gets lower cost but lower accuracy*
un-pruned*	yes		yes	yes	10.88	*Never gets lower cost than EG2, (-0.007), accuracy always lower.*
breast	**can combine strategies to improve results*					
pruned*	no	3.74	yes	yes	11.57	*EG2 gets lower cost but lower accuracy.*
un-pruned	no	0.65	yes	yes	8.48	*Does not require combination of strategies to achieve its aim.*
car	*unable to meet aims regardless of combinations of strategies*					
pruned	no	14.01	no	yes	33.56	
un-pruned	no	16.28	no	yes	35.83	
diabetes	*does not need any combination of strategies to meet aims*					
pruned	yes		yes	yes	13.83	
un-pruned	no	0.08	yes	yes	14.08	
flare	*does not need any combination of strategies to meet aims*					
pruned	yes		yes	yes	4.6	
un-pruned	yes		yes	yes	2.04	
glass	**can combine strategies to improve results*					
pruned*	yes		yes	yes	21.9	*EG2 has low cost but accuracy not as high.*
un-pruned*	yes		yes	yes	24.01	*Combinations of strategies required*
heart	*does not need any combination of strategies to meet aims*					
pruned	yes		yes	yes	10.2	
un-pruned	yes		yes	yes	11.42	
hepatitis	*does not need any combination of strategies to meet aims*					
pruned	no	0.06	yes	yes	27.31	
un-pruned	yes		yes	yes	27.25	
iris	**can combine strategies to improve results*					
pruned*	yes		no	yes	46.94	*Combinations of strategies help, but there is some sacrifice*
un-pruned*	yes		no	yes	46.94	*Combination of strategies help meet the aims*
krk	*unable to meet aims regardless of combinations of strategies*					
pruned	no	23.53	yes	no		
un-pruned	no	25.95	yes	yes	39.9	
mushroom	*lowest cost and highest accuracy already used, no combination possible*					
pruned	yes		yes	no		*All aims achieved, EG2 reduces cost by 0.003*
un-pruned	yes		yes	no		*All aims achieved, EG2 reduces cost by 0.002*
nursery	*unable to meet aims regardless of combinations of strategies*					
pruned	no	20.85	no	yes	47.2	
un-pruned	no	22.05	no	yes	48.4	
soybean	*lowest cost and highest accuracy already used, no combination possible; STRATEGY0 similar results to STRATEGY1*					
pruned	no	0.73	no	no		*Difference in costs only on average 0.0076.*
un-pruned	no	0.96	no	no		*Difference in costs only on average 0.0032.*
tictactoe	*lack of cost information for some CMs is hard to overcome*					
pruned	yes		yes	yes	22.34	*Can meet aims, performs better than all except ACT.*
un-pruned	no	1.59	yes	yes	24.22	*As above*
wine	**can combine strategies to improve results*					
pruned*	no	0.01	yes	yes	14.7	*All algorithms sacrifice accuracy for lower costs*
un-pruned*	no	0.56	yes	yes	16.74	*As above*

Table 16 Summary of whether MA_CSDT has met its aims for each dataset

The first aim has been met for the majority of datasets. For all other datasets where a sacrifice of accuracy rate has been required in order to return a lower cost than J48, this sacrifice has been kept to a minimum. The lowest cost over all algorithms has been returned for all datasets except annealing (pruned), krk (pruned), soybean and mushroom; where these costs are lower there is a large amount of sacrifice of accuracy rate. For these latter two datasets, all algorithms have returned low costs and the differences between all these costs are very small and all strategies produce very similar values.

The costs for each dataset returned by each algorithm, both pruned and un-pruned versions, have been averaged over all the cost matrices and for the multi-class datasets, also over the 3 groups of misclassification costs; mixed, low and high. Graphs have been produced which show the average cost returned by each algorithm and by each MA_CSDT strategy and annotated with the corresponding average accuracy rate obtained. The graphs have been examined and have been divided into three categories; those where the MA_CSDT algorithm has achieved its aim of returning a high accuracy rate or minimal sacrifice to the accuracy rate in a more cost-effective way than J48 and the other cost-sensitive algorithms, those where it has not achieved its aim, and those where the aim could be achieved if an increase in the sacrifice to the accuracy rate is allowed. A representative dataset has been chosen to illustrate the findings of the evaluation, with the rest of the graphs presented in Appendix Sections D5 and D6 for pruned and un-pruned versions respectively with Appendix Sections D7 and D8 which have tables showing the results obtained for each cost matrix by each strategy for costs and accuracy for pruned and un-pruned versions respectively.

The following sections describe the results in each of the three scenarios; Section 6.1.1 presents the results where the aim of the algorithm has been met using a consistent strategy

throughout the cost matrices for a dataset, Section 6.1.2 presents the results from those datasets where the aim of the algorithm has not been met regardless of how flexible the algorithm can be and Section 6.1.3 presents the results where the aim can be met by choosing the most appropriate strategy for each cost matrix for a dataset.

6.1.1 Results from the evaluation where the aim has been achieved by MA_CSDT

The algorithm MA_CSDT has achieved its aim on datasets heart, flare, hepatitis, diabetes, tic-tac-toe (pruned) and breast (un-pruned). Figure 21 presents the results for the heart dataset, showing the cost averaged over all cost matrices.

The algorithm J48 returns an accuracy rate of 75.75% and obtains this accuracy rate at a cost of 0.296. All other algorithms return a lower cost than this. The highest accuracy rate returned is 78.42% and the algorithm MA_CSDT using ACCURACY-BASED(ALL) produces this value. The accuracy rate has been increased by 2.67% above that returned by J48 at a reduction in cost of 0.066. Using ACCURACY-BASED<TRUE>, although the increase in accuracy is lower at 1.21%, the reduction in cost is higher at 0.087. As shown in Table 14, *AdaCostM1* and *MetaCost*, produce the fewest trees for this dataset and as a consequence, the accuracy rate returned by these algorithms are much lower. *ACT* grows 93% of the trees, whilst this is enough to increase the accuracy rate of this algorithm against *AdaCostM1* and *MetaCost*, the cost returned is increased owing to the increased misclassification costs.

The algorithm MA_CSDT using ACCURACY-BASED<TRUE> returns a higher accuracy rate than *EG2* and a lower cost. For the un-pruned versions of the algorithms, J48 returns an accuracy rate of 76.97% at a cost of 0.343.

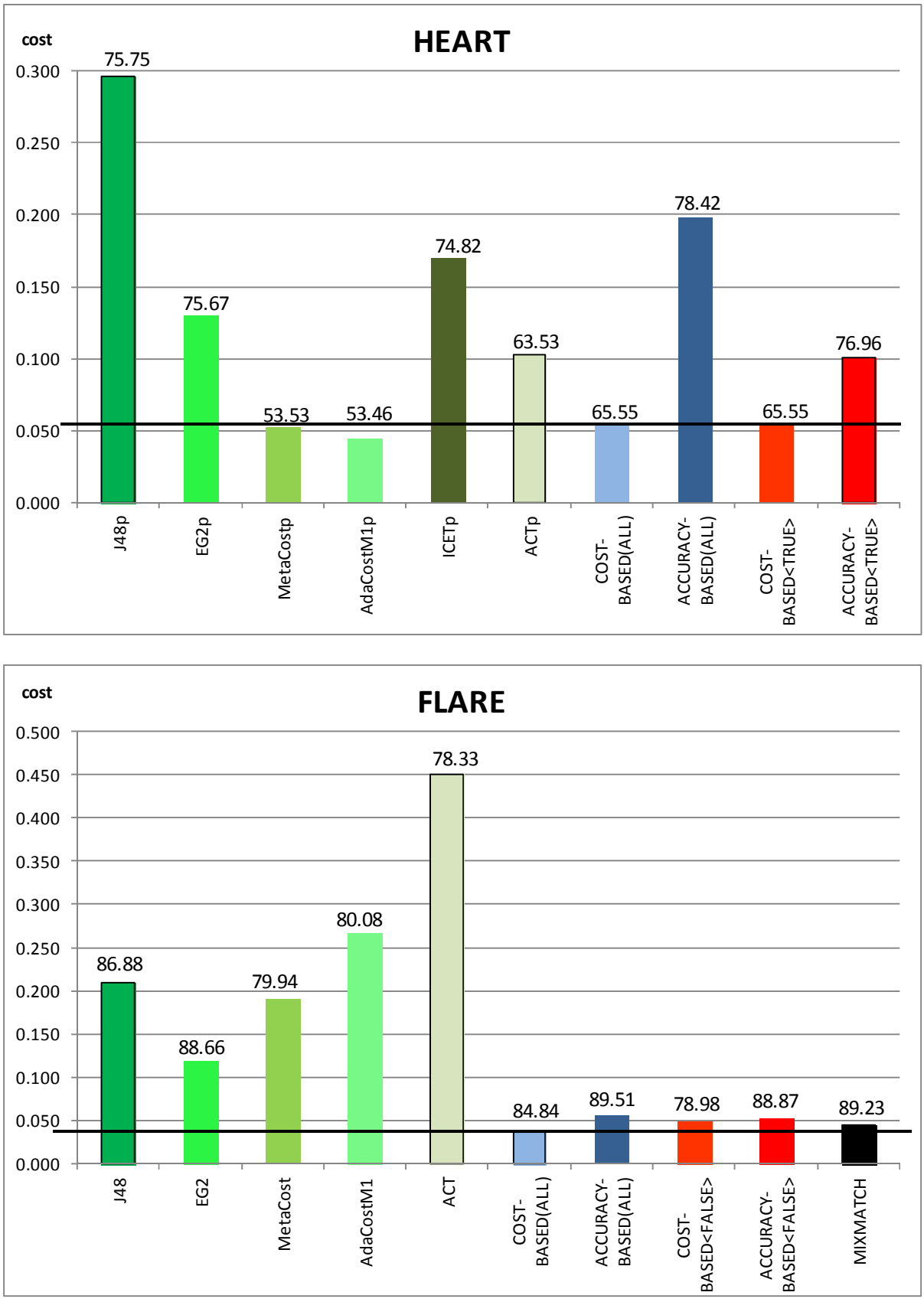


Figure 21 Heart dataset processed using pruned versions of the cost-sensitive algorithms and flare dataset using un-pruned versions of the cost-sensitive algorithms

In the case of the flare dataset, each of the other algorithms fails to grow trees for some of the cost matrices. In particular *EG2* does not grow any trees at all, *ICET* does not grow trees 84% of the time and even J48 does not grow any trees 70% of the time. After careful examination of the output files produced, it has been concluded that the lack of trees are as a result of the class distribution of the dataset which causes the majority of trees to be pruned back to nothing. The majority class in the whole dataset has 88.9% of the examples in it. Pruning techniques would most likely determine that sub-trees were not able to improve on the results of the original dataset and so would be converted into leaves, resulting in a large percentage of no models being produced. With this in mind, the results of the flare dataset using un-pruned versions of the algorithms have been presented in Figure 21.

The MA_CSDT algorithm using all strategies can be forced to induce trees using its parameter settings, setting the pre-pruning option to 'd' and by allowing no proportional stopping. The ACCURACY-BASED(ALL) and ACCURACY-BASED<FALSE> return an accuracy rate increase of 0.07% and 0.01% respectively and manages to return a lower cost than J48.

The accuracy rate for J48 is 86.88%, for a cost of 0.209. Of the cost-sensitive algorithms, *AdaCostM1* and *ACT* do not return as lower a cost as J48, as the accuracy rates obtained by these two algorithms are very much reduced, an indication of how no model was produced using pruned versions of these algorithm as the leaf would have had less errors than the sub-tree. Of the five existing cost-sensitive algorithms, *EG2* returns the lowest cost and increases accuracy by 1.78%. This is a good result, however, MA_CSDT using ACCURACY-BASED<FALSE> returns a similar increase but for 0.067 less cost. The ACCURACY-BASED(ALL) accuracy rate is an increase of 2.63% over J48 and 0.85% over *EG2*. As

illustrated in Figure 21 each strategy of MA_CSDT returns a lower cost than the other six algorithms.

6.1.2 Results from the evaluation where the aim has not been achieved by MA_CSDT

The algorithm MA_CSDT has not achieved its aim on datasets car, nursery and krk. Figure 22 presents the results for the krk dataset, which is representative of this group of datasets.

No algorithm returns a higher accuracy rate than J48, although it accomplishes this at a greater cost than each of the other algorithms including all strategies of MA_CSDT. *MetaCost* and *AdaCostM1* get closer to the accuracy rate of J48 than any other algorithm and return a lower cost than J48. MA_CSDT does not get anywhere near this accuracy rate, the highest accuracy rate is produced by ACCURACY-BASED<TRUE> and does return a low cost but this accuracy rate is 23.53% less than J48.

Using un-pruned versions, the results for *EG2*, *ACT* and all strategies of MA_CSDT show no improvement and in most cases are worse with regards to the cost value. J48 has a higher accuracy and slightly lower cost, *MetaCost* and *AdaCostM1* remain the better algorithms regarding returning the highest accuracy rates at a slight reduction in cost. The *ACT* algorithm returns the highest cost and the lowest accuracy of all the algorithms. The MA_CSDT using ACCURACY-BASED<FALSE> in the un-pruned version has higher cost than in the pruned version and the accuracy rate is reduced.

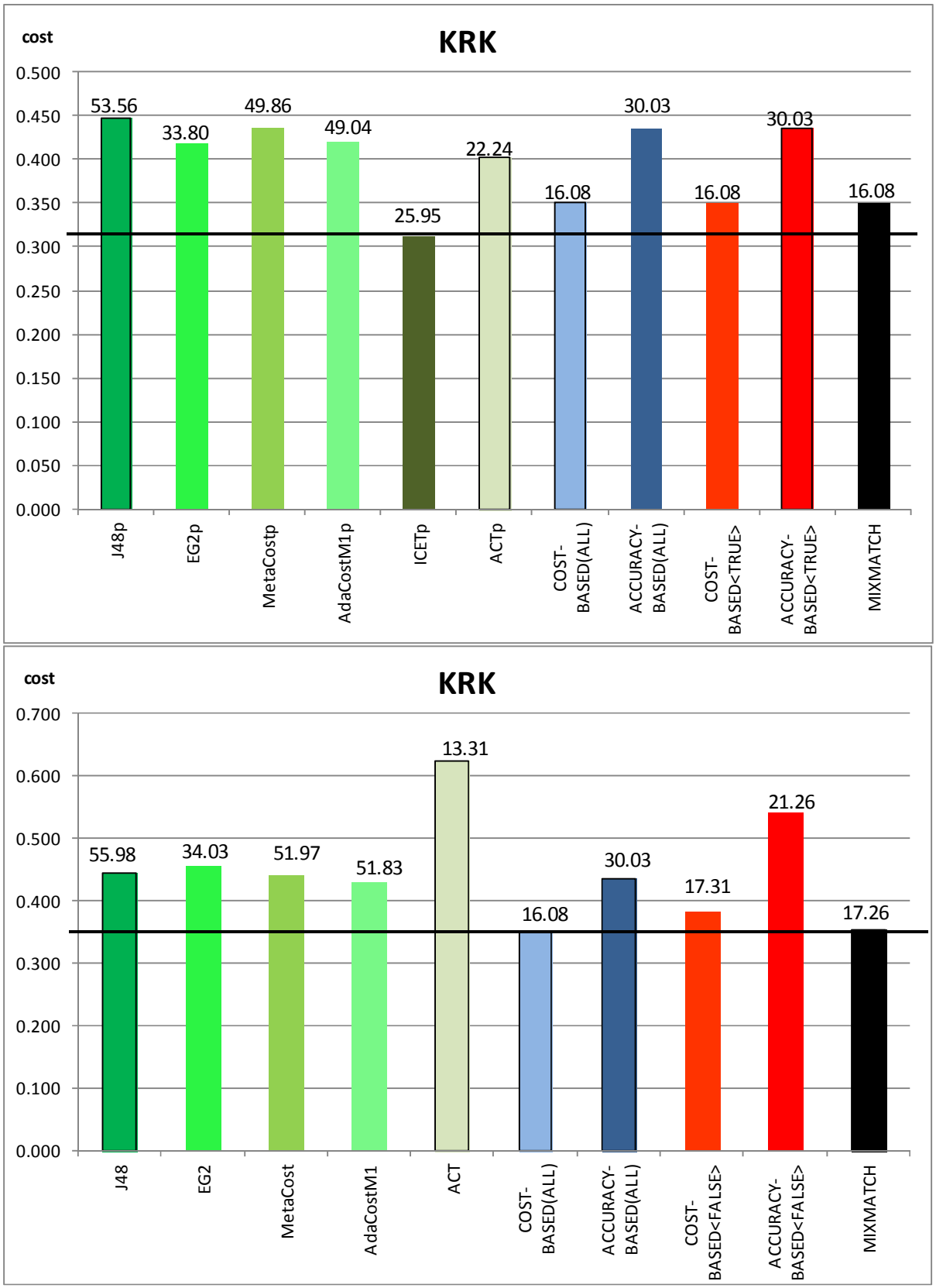


Figure 22 The krk dataset: top processed using pruned version; bottom processed using un-pruned version of cost-sensitive algorithms

6.1.3 Results from the evaluation where the aim can been achieved by MA_CSDT

Although the algorithm MA_CSDT has not quite achieved its aim on some datasets, it has almost achieved it with perhaps one cost-sensitive algorithm producing slightly better results or perhaps not quite reaching a high accuracy. Figure 23 presents the results for the iris dataset, with the cost obtained from each algorithm and MA_CSDT strategies. This dataset is representative of the datasets where using the mentioned four strategies or a mixture of them do not always improve on the results; annealing, breast, glass, tic-tac-toe and wine.

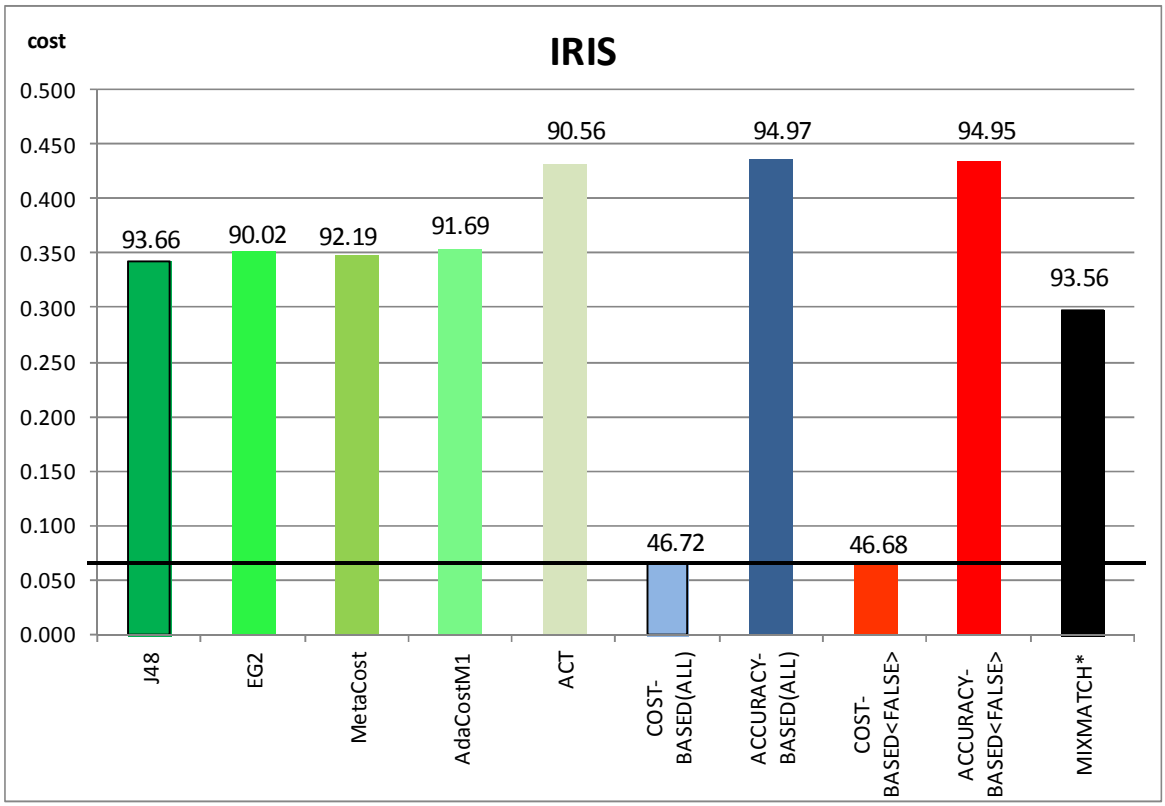


Figure 23 Iris dataset processed using the un-pruned version of the cost-sensitive algorithms

An examination of the cost and accuracy rates obtained from the described strategies show that the algorithm MA_CSDT using COST-BASED<FALSE> produces the lowest cost of all the algorithms and strategies, however this results in a huge sacrifice of the accuracy rate which is reduced by 46%. The ACCURACY-BASED(ALL) and ACCURACY-

BASED<FALSE> return the highest accuracy rates and improve over J48 by 1.3%, but the costs returned by these two strategies is far too high and is an increase on the cost returned by J48 and the cost-sensitive algorithms. The cost-sensitive algorithms return a cost that is in each case higher than J48.

Owing to the fact that the MA_CSDT algorithm has multiple parameter settings which can be used in different combinations, it is possible to examine results obtained for this dataset and each cost matrix for all possible parameter settings in order to find a result which improves upon the results obtained by the five cost-sensitive algorithms.

The MA_CSDT MIXMATCH strategy as described, involves looking at the four other strategies and choosing from these four strategies, the best for each individual cost matrix. For the iris dataset, there is no variety in the four strategies, both the cost-based values and the accuracy-based values are the same. However, there are other experimental results which could potentially be better than those examined, which were the highest accuracy or lowest cost.

As mentioned, the accuracy rate returned by both accuracy-based strategies are 94.97% and 94.95%. This is higher than the J48 accuracy rate of 93.66%. Therefore all experimental results for the iris dataset have been examined for an accuracy rate which is not the highest but is close to the highest whilst returning a cost which is lower than the other cost-sensitive algorithms.

Table 17 presents the cost and accuracy returned for individual cost matrices for the iris dataset for each algorithm, and for each strategy of MA_CSDT. The entry MIXMATCH*

shows values returned when all experiments for the iris dataset are examined looking at the highest accuracy obtained when the cost is lower than that obtained by other algorithms.

For each cost matrix, it is possible to find a combination of parameter settings which can obtain a good accuracy rate but do this in a more cost-effective way.

cost	1	2	3	4	5	6	7	8	9
J48	0.342 ± 0.016	0.329 ± 0.016	0.336 ± 0.017	0.421 ± 0.022	0.42 ± 0.021	0.421 ± 0.022	0.277 ± 0.013	0.27 ± 0.013	0.273 ± 0.014
EG2	0.342 ± 0.008	0.337 ± 0.008	0.342 ± 0.009	0.419 ± 0.011	0.421 ± 0.011	0.419 ± 0.011	0.293 ± 0.008	0.291 ± 0.008	0.292 ± 0.008
MetaCost	0.358 ± 0.005	0.353 ± 0.007	0.328 ± 0.01	0.448 ± 0.023	0.472 ± 0.018	0.397 ± 0.023	0.279 ± 0.013	0.263 ± 0.009	0.246 ± 0.01
AdaCostM1	0.37 ± 0.001	0.36 ± 0	0.29 ± 0.001	0.47 ± 0.017	0.461 ± 0	0.373 ± 0.001	0.325 ± 0.01	0.274 ± 0.011	0.257 ± 0.015
ACT	0.42 ± 0.01	0.413 ± 0.011	0.42 ± 0.01	0.526 ± 0.013	0.524 ± 0.013	0.526 ± 0.013	0.36 ± 0.01	0.351 ± 0.009	0.35 ± 0.01
COST-BASED(ALL)	0.014 ± 0	0.014 ± 0.001	0.013 ± 0.001	0.01 ± 0	0.01 ± 0	0.009 ± 0	0.178 ± 0.012	0.149 ± 0.009	0.175 ± 0.012
ACCURACY-BASED(ALL)	0.5 ± 0.014	0.428 ± 0.011	0.45 ± 0.017	0.504 ± 0.014	0.5 ± 0.017	0.614 ± 0.027	0.305 ± 0.008	0.38 ± 0.009	0.245 ± 0.01
COST-BASED<FALSE>	0.014 ± 0	0.014 ± 0.001	0.013 ± 0.001	0.01 ± 0	0.01 ± 0	0.009 ± 0	0.179 ± 0.012	0.149 ± 0.009	0.176 ± 0.012
ACCURACY-BASED<FALSE>	0.5 ± 0.014	0.428 ± 0.011	0.45 ± 0.017	0.504 ± 0.014	0.5 ± 0.017	0.614 ± 0.027	0.305 ± 0.008	0.38 ± 0.009	0.245 ± 0.01
MIXMATCH*	0.3 ± 0.009	0.28 ± 0.008	0.289 ± 0.01	0.365 ± 0.007	0.369 ± 0.011	0.357 ± 0.011	0.243 ± 0.005	0.236 ± 0.004	0.239 ± 0.008

accuracy	1	2	3	4	5	6	7	8	9
J48	93.66 ± 0.767	93.66 ± 0.767	93.66 ± 0.767	93.66 ± 0.767	93.66 ± 0.767	93.66 ± 0.767	93.66 ± 0.767	93.66 ± 0.767	93.66 ± 0.767
EG2	90.02 ± 1.456	90.02 ± 1.456	90.02 ± 1.456	90.02 ± 1.456	90.02 ± 1.456	90.02 ± 1.456	90.02 ± 1.456	90.02 ± 1.456	90.02 ± 1.456
MetaCost	94.42 ± 0.931	94.42 ± 0.931	78.53 ± 4.015	93.11 ± 0.785	94.61 ± 0.948	93.65 ± 0.792	93.42 ± 0.751	93.66 ± 0.767	93.9 ± 0.845
AdaCostM1	90.78 ± 1.217	94.67 ± 0.846	78.53 ± 4.015	93.95 ± 0.928	94.67 ± 0.846	90.7 ± 1.468	94.14 ± 0.885	93.66 ± 0.767	94.11 ± 0.749
ACT	87.12 ± 1.664	87.59 ± 1.47	89.41 ± 1.228	90.52 ± 1.267	91 ± 1.256	89.41 ± 1.228	92.95 ± 0.763	92.95 ± 0.763	94.11 ± 0.75
COST-BASED(ALL)	33.81 ± 1.45	31.38 ± 1.452	34.81 ± 1.306	33.81 ± 1.45	31.38 ± 1.452	34.81 ± 1.306	73.03 ± 2.228	71.86 ± 2.038	75.62 ± 2.046
ACCURACY-BASED(ALL)	94.62 ± 0.9	95.33 ± 0.66	94.75 ± 0.771	95.12 ± 0.791	94.7 ± 1.076	94.7 ± 0.89	95.26 ± 0.904	95.04 ± 0.79	95.22 ± 0.537
COST-BASED<FALSE>	33.81 ± 1.45	31.38 ± 1.452	34.81 ± 1.306	33.81 ± 1.45	31.38 ± 1.452	34.81 ± 1.306	72.39 ± 2.208	71.86 ± 2.038	75.86 ± 1.893
ACCURACY-BASED<FALSE>	94.62 ± 0.9	95.33 ± 0.66	94.75 ± 0.771	95.12 ± 0.791	94.7 ± 1.076	94.7 ± 0.89	95.26 ± 0.904	95.04 ± 0.79	95.22 ± 0.537
MIXMATCH*	92.44 ± 1.126	92.73 ± 1.023	93.61 ± 1.054	93.34 ± 1.07	93.18 ± 1.045	92.26 ± 1.4	94.75 ± 0.771	94.75 ± 0.771	94.99 ± 0.654

Table 17 Iris dataset cost and accuracy returned for each cost matrix for un-pruned versions of cost-sensitive algorithms

The MIXMATCH* bar in Figure 23 shows the improvement possible when examining other parameter settings other than those producing the highest accuracy overall. By accepting a reduction in the accuracy rate of 0.01%, it is possible to reduce the cost by 0.045. The *EG2* algorithm returned a higher cost than J48 for a much lower accuracy so the improvement obtained by the MA_CSDT using MIXMATCH* strategy is 0.053 lower cost for an increase of accuracy of 3.54%.

6.2 Discussion of the outcome of the empirical evaluation

The MA_CSDT algorithm can return the lowest cost for a cost matrix 68.2% of the time and return the highest accuracy for a cost matrix 34.98% of the time. Each time the highest accuracy is achieved, its corresponding cost is lower than that of J48. The main aim to achieve the same or higher rate of accuracy more cost-effectively as the accuracy-based algorithm J48 has been met for the datasets annealing, flare, glass, iris, heart and mushroom. For the datasets diabetes, hepatitis, tic-tac-toe and wine a sacrifice of less than 1% of the accuracy rate returned by J48 results in a lower cost. The un-pruned version processing the breast dataset also returns a lower cost for the same sacrifice but for the pruned version the sacrifice was greater at 3.74%. For the remaining datasets, car, krk, nursery and soybean, this aim has not been met.

By looking at the two strategies cost-based and accuracy-based it is apparent that a trade-off is required between cost-based decisions i.e. cost-based strategy choosing the parameter setting combination which produces the lowest cost for a cost matrix and accepting the corresponding accuracy rate, and accuracy-based decisions i.e. accuracy-based strategy choosing the parameter setting combination which produces the highest accuracy rate for a cost matrix and accepting the corresponding cost. As discovered in Chapter 5, the same parameter settings seldom produce both the lowest cost and the highest accuracy so it is evident that all processes of classification is a trade-off; attribute selection and the decision of which model to choose which would produce the desired result for user requirements and domain requirements.

The heart dataset is representative of six datasets which produce good results and achieve the aim of the algorithm by obtaining a high accuracy rate in a more cost-effective way. For each

of the datasets where the aim of the MA_CSDT algorithm has been met, these good results are produced by using the accuracy-based strategy. The cost-based strategy has the lowest cost which has been returned from induced trees; however the corresponding accuracy rates are always lower than the J48 algorithm. In the datasets where the aim has been met, choosing the same strategy for each of the cost matrices has been sufficient to meet the aim of MA_CSDT, thus a compromise between accuracy-based decisions and cost-based decisions has been found.

The krk dataset is representative of poor results, where the MA_CSDT algorithm has not been able to achieve its aim. The accuracy-based algorithm J48 achieved the highest accuracy overall, with only two of the cost-sensitive algorithms *MetaCost* and *AdaCostM1* achieving a similar accuracy rate. They achieved almost the same rate with a small reduction in costs. *EG2*, *ICET* and *ACT* also failed to achieve a comparable accuracy rate but do manage to at least reduce costs.

On examination of the trees induced, the most likely cause of this failure to meet its aim is that the MA_CSDT algorithm either grows trees that are too small in comparison with the size of the dataset, which has a large number of examples in the training set, or grows a tree which is far too large with over 20,000 leaves. The paths in these trees are comprised of 6 attributes in order to reach the leaves. The smaller trees have paths to the leaves which consist of two or three attributes. The conclusion reached, after studying the files produced is that the large number of examples contributes to the problems of processing this dataset. In each training file there are approximately 20,000 examples. Pruning or stopping tree build early results in many examples at the leaf nodes which results in low accuracy. Allowing the tree to grow fully, results in overgrown trees with few examples at the node, which results in

higher test costs but still does not improve accuracy. The *ACT* algorithm also produces trees which follow this pattern of either too small or too large and is also not very successful on this dataset.

The three algorithms, which produce the better results for this dataset (J48, *MetaCost*, *AdaCostM1*), all produce the same tree which, when pruned has 3948 leaves and un-pruned 8268 leaves. They all choose the same root; 'wkr' which happens to be the attribute with the highest cost. However this is statistically the better attribute and in combination with less costly attributes further down the tree, results in a medium sized tree which produces the accuracy rate of around 50%. *EG2* produces similar trees but chooses one of the less costly attributes for its root, and uses the more costly one later in the tree build. This produces a similar sized tree but the less costly attribute is not as good a root attribute as the more costly one in this dataset. *ACT* chooses the same root as the better algorithms, however in its pruned version, reduces the tree produced too much and in the un-pruned version grows a tree with over 25,000 leaves, increasing the test costs needed as a result of later choosing many less costly attributes.

This dataset demonstrates that higher accuracy does not always mean lower costs, if to achieve this, the tree grows uncontrollably. However not spending a lot on test costs does not always work either as money saved on tests may be spent on misclassification costs.

On examination of the results of the soybean dataset, all of the algorithms except *ACT* produce very similar results. All the accuracy rates are slightly lower than J48 and the costs returned by all strategies are around the same value and all algorithms return higher costs than J48. This dataset is one of the few where the cost-based strategy and the accuracy-based

strategy produce very similar results. It is likely that this is as a result of low test costs with many attributes having the same test costs. It is one of the datasets where there are parameter settings which produce both the lowest cost and the highest accuracy as investigated in Chapter 5. The tic-tac-toe dataset is a difficult dataset to process for a cost-based algorithm in that as the test costs for every attribute is the same value, the misclassification costs become the only source of information for MA_CSDT to use in order to determine the attributes to use in the tree build. When processing particular cost matrices where the misclassification costs are low or are the same as the test costs, there is little information that can be used in order to separate the attributes into useful ones.

Examining the results of the mushroom dataset demonstrates the successful trade-off between the test costs and misclassification costs. Each algorithm, for either all cost matrices or some cost matrices, can achieve 100% accuracy. The costs obtained are the test costs used in the tree induction. Any variation between algorithms regarding the cost they return are the sole result of the attributes which have been chosen during tree induction and therefore when different trees are induced this can be inferred by looking at the cost obtained. Therefore the objective of the algorithm is to discover the best attributes, which in combination can achieve the 100% accuracy but incurring the smallest amount of test costs. Of all the algorithms in the evaluation, *EG2* returns the lowest overall cost with a corresponding accuracy of 100%. The MA_CSDT returns a slightly higher cost, but this is around 0.002 to 0.003 higher than *EG2* (which is helped by its formulae in this situation but not for other datasets with similar ranges of test costs) but achieves one aim of returning a lower cost than the accuracy-based algorithm and so has successfully found a trade-off between the two viewpoints.

As demonstrated in Chapter 5, examining the parameter settings using the cost-based strategy and the accuracy-based strategy, these are, more often, not the same parameter settings for the same cost matrix. The user is given a choice between the two viewpoints which will be determined by the domain nature. For example, some domains may like to err on the side of caution. They prefer to reduce costs and are not concerned with the reduction of the accuracy rate. An example would be checking for fraudulent bank transactions. In this case if a bank transaction is checked and is not fraudulent the cost may simply be the cost of a telephone call. If the transaction is actually fraudulent then the cost saved could potentially be quite high. Therefore the bank would not be overly concerned by false positive errors as they would not incur high misclassification costs.

Other domains are opposite. They do not want high cost errors but do not want the low cost errors either. Medical domains are examples of this. They do not want to subject the patient to unwanted treatment either so in this case they are more likely to want a classifier which will be accurate but obviously if they can obtain this accuracy at a lower cost this is much more desirable.

With some datasets, choosing a straightforward accuracy-based strategy for each cost matrix for a dataset is enough to obtain a higher accuracy rate for a lower cost. For some datasets, choosing between different strategies for each cost matrix improves on the results overall. For some datasets this is not always possible as the four strategies produce only 2 different results. Quite often the accuracy rate using the cost-based strategy is too low and there is too much sacrifice. However it may be possible to examine the results and not choose the highest accuracy but get a lower cost. In some cases other parameter settings may produce high accuracy rates which may not be lower than J48 so the aim of the algorithm is still met, thus

taking advantage of the flexibility of the MA_CSDT algorithm. In other cases this may result in sacrifice of the accuracy rate but in acceptable amounts. For example when *AdaCostM1* or *MetaCost* process the heart dataset the accuracy rate they produce is over 30% less than J48, this sacrifice is far too much. A reduction of around 3% for a significant reduction in costs is acceptable.

In the iris dataset, the cost-based strategy returns low cost but the corresponding accuracy rate is far too low; the accuracy-based strategy returns higher accuracy than J48 but for a higher cost. The cost-sensitive algorithms return a lower cost with a more comparable accuracy rate. The likely reason for this scenario is that where pruned versions are used, a sub-tree was removed which reduced the errors on the training set but resulted in the accuracy rate obtained on the testing set which could potentially have been higher if not pruned, but is still comparable. As the tree was smaller, there was a reduction in test costs used which ultimately resulted in a lower cost. In the un-pruned version J48 can take advantage of its statistical measure. As there are only four attributes in the dataset with two with much lower test costs than the other two, it is likely that growing the tree is the ideal way to increase accuracy but it is harder to control the tree build with regard to cost-effectiveness owing to the small number of attributes in the dataset.

However, the purpose of MA_CSDT is to try to find a compromise between such situations as these. So it makes sense to examine other parameter settings to find a compromise so that the accuracy rate can be increased in a more cost-effective way. The MIXMATCH* strategy is designed to replicate the user's decisions, taking into account the domain nature so that it can be demonstrated that the MA_CSDT's flexible nature can be exploited. In the case of the iris dataset, the MIXMATCH strategy has been extended to include all results which have

been obtained by all combinations of parameter settings in order that MA_CSDT's best and comparable performance can be utilized in the comparison.

As demonstrated in the representative dataset krk, the overall accuracy rate is nowhere near as high as the J48 algorithm so therefore no amount of strategy mixing is going to help. It could be that just going for a lower cost is the only alternative. This may be acceptable in some domains as mentioned so if the algorithm returns the lowest cost of all of the algorithms then maybe this would be an alternative for the user. In this dataset, it might be the case that low costs are hiding bad splits and the only way to help is to try to gain additional information by some other means. If there is a lot of activity regarding cost i.e. high test costs, high misclassification costs or a mixture, the cost information is enough to construct the tree. If the reverse is the case as in tic-tac-toe dataset or krk where there are 50% very low cost attributes and 50% very high cost attributes, maybe a statistical measure is required to add that little bit extra knowledge. What may be useful is the reverse of those algorithms described in Section 3.1.1.1 where the statistical measure has been extended to include cost, maybe the cost calculation used in the MA_CSDT can be extended to include a statistical measure, when confronted by these combinations of test costs and misclassification costs in a dataset. When the test cost information is minimal as in tic-tac-toe this might also be a good addition because there is only the misclassification cost to use. It may take a while to get to good misclassification cost reduction so tree induction results in a large tree. Even though the cost of the tests may not be too much it all adds up and an overlarge tree tends not to be helpful in most cases, hence the introduction of pruning techniques discussed by Frank and Witten (1998). Balancing the reduction in misclassification costs against a small output of test costs does not usually work all that well but testing whether the attribute is good for splitting using another measure in this case would probably be best.

As mentioned with the soybean dataset, because many of the combination of parameter settings result in both low costs and high accuracy, there is also not much room for flexibility by mixing the strategies. In order to increase accuracy to that of the accuracy-based algorithm J48 it might also require some additional information which is provided by a statistical measure. The mushroom dataset has also a range of low test costs. Although MA_CSDT can produce the second lowest costs and the differences between the two algorithms' low cost is very small, there are several attributes with the same test cost, some of which are more useful than others. In order to find which one is the better split when the cost is identical, the information provided by a statistical measure may be useful, and the algorithm which does produce the lower costs for the mushroom dataset, *EG2*, does use statistical measures along with the test costs.

The *EG2* equation balances the amount of information gained against the test cost spent to achieve this information. If the cost is worth it regarding the information gained this is reflected in the value returned and so therefore the highest one is chosen, indicating good information for cost ratio. Although nothing is done if the attributes are not worth the cost maybe something like this could be used with that described in Section 4.3 where the 'worth' of the choice of split is determined. It could be included with the test costs and misclassification costs, in the datasets where it is needed, to determine whether the low cost is hiding a bad split or is simply a low cost attribute that will produce a good split.

Although not successful in every dataset, there is sufficient evidence to suggest that it is possible to find a compromise between accuracy-based decisions and cost-based decisions in order to both maintain accuracy and return lower costs or to minimize the sacrifice of the accuracy rate whilst still returning lower costs.

6.3 Summary of the findings of the evaluation

In order to examine the Multi-Armed Cost-Sensitive Decision Tree algorithm (MA_CSDT) and compare performance with well-known existing cost-sensitive algorithms, 15 datasets along with test costs and misclassification costs have been used in a comparison along with J48. The aim of the algorithm is to minimize costs whilst maximizing the accuracy rate. The J48 algorithm provides the target accuracy rates. The existing cost-sensitive algorithms are *EG2*, *MetaCost*, *AdaCostM1*, *ICET* and *ACT*.

The main aims are that the MA_CSDT algorithm returns the same accuracy rate or higher than J48, that this is obtained at a lower cost and that it produces the lowest cost than the other algorithms. For the majority of datasets these aims have been met.

For some datasets, using a single strategy for all cost matrices obtain good results, other datasets obtain good results when using a mixture of strategies for the cost matrices and there are three datasets; car, krk and nursery where no strategy has been able to produce accuracy rates comparable with the accuracy-based algorithm J48.

If there is a lot of activity regarding cost i.e. high test costs, high misclassification costs or a mixture of high and low costs, the cost information is enough to construct the tree. If the reverse is the case it is difficult to construct the tree as there would only be one cost element contributing to the information needed. If, as in the case of the tic-tac-toe dataset, for one of the cost matrices there is minimal cost information as the test costs are all minimal and so is the misclassification cost, maybe a statistical measure is required in order to supplement the information provided by the costs.

The algorithm can use its parameter settings and differing strategies to overcome the problems caused by difficult datasets. There is sufficient evidence to suggest that it is possible to find a balance between the cost-based decisions and accuracy-based decisions in order to meet the aim of minimizing cost and maximizing the accuracy or to minimize sacrifice of the accuracy rate.

CHAPTER 7: CONCLUSIONS AND FUTURE WORK

Cost-sensitive decision tree learning is an important research area as it considers the costs involved when inducing decision trees. In the real world costs are involved when obtaining data and when classification errors occur. Recent comparisons have evaluated algorithms which have incorporated these costs by various methods such as extensions to statistical measures, genetic algorithms or boosting and bagging techniques. Examining the results of the better performing algorithm revealed some weaknesses.

This thesis has suggested that cost-sensitive decision tree learning involves a trade-off between decisions based on accuracy and decisions based on costs and that Game Theory can be utilized to develop a framework which can find a compromise between these two points of view. The aim of the thesis is to demonstrate the trade-off between the accuracy-based decisions and the cost-based decisions and use the trade-off effectively to achieve the aim of cost-sensitive learning which is to minimize costs whilst maximizing accuracy. The nature of the domain dictates the importance of this aim. Whilst some domains may err on the side of caution and prefer to sacrifice the accuracy rate rather than incur high misclassification costs, there are many domains where this is not acceptable. Medical domains would prefer to have no low cost errors either, not wanting to subject the patient to unwanted treatment. In these domains, if a classifier can be found which will minimize costs, but at the same time be as accurate as an accuracy-based classifier, this is more desirable.

A number of methodologies were examined and the category Nomothetic methods chosen as the more appropriate for this research. A method named GQM (Goal, Question, Metric) (Basili and Weiss 1984) has been used for this research. In this thesis, the goal was to develop a framework which uses the trade-off required between accuracy and costs in order

to achieve low costs and high accuracy required in cost-sensitive learning. This goal was defined in order that the hypothesis might be proven. The questions raised by this were (i) how well do existing cost-sensitive decision tree algorithms perform? (ii) What are the weaknesses of existing cost-sensitive decision tree algorithms? (iii) Is it possible to minimize costs and minimize the sacrifice of the accuracy rate which occurs in cost-sensitive decision tree learning? and (iv) Will using a technique which has been developed to deal with trade-off by using pay-offs help in achieving the aim of cost-sensitive decision tree learning?

These questions help to devise the objectives needed to focus the research, carrying out a literature review to find answers. Metrics were also devised which measure cost to determine whether these costs are minimized and accuracy, to determine whether this is maximized or that the sacrifice is minimized.

It was felt that as the GQM methodology has been primarily aimed at software development and that a main part of this thesis is to develop a new framework for cost-sensitive decision tree learning, it would be the better methodology to follow. It has been concluded that this methodology was helpful in providing a good interface between theory and practice and that of all the methodologies investigated, was the most suited to the task. It is helpful to define a goal which can be related to the research hypothesis, using the questions which can give rise to good objectives and allowing a practical approach to the research.

The following objectives were devised in order to achieve the answers to the questions devised using the GQM methodology and to meet the aims of the thesis using the metrics given:

1. To survey and review existing cost-sensitive decision tree algorithms in order to investigate ways in which costs have been introduced into the decision tree learning process and at which stages they have been introduced
2. To evaluate existing cost-sensitive decision tree algorithms in order to discover whether these algorithms are successful over many types of problems or are only effective for some types of problems, for example binary class datasets or balanced datasets
3. To develop a new cost-sensitive decision tree algorithm which is based on Game Theory
4. To investigate and evaluate the new algorithm against existing algorithms and measure performance in terms of cost to classify and accuracy, in order to test the research hypothesis

Some recent comparisons show that although many cost-sensitive algorithms can process balanced, two class datasets well, many of them produce lower accuracy rates in order to achieve lower costs to classify when the dataset is less balanced or has multiple classes.

One comparison has been examined further and the results of the algorithm which proved the better one, has been analyzed to determine the weaknesses of cost-sensitive decision tree algorithms. The main weaknesses discovered are (i) problems arise from an imbalance in the class distribution, (ii) multi-class datasets cause problems, (iii) extreme misclassification costs are difficult to handle and (iv) trade-off between high misclassification costs usually result in the accuracy rate being sacrificed; the higher the misclassification costs, the more unbalanced the class distribution, the lower the accuracy rate tends to be.

As part of an extensive literature review, a survey of cost-sensitive decision tree algorithms has been carried out revealing over 50 different algorithms. A timeline of these algorithms is presented. The algorithms have been organized by the way costs have been introduced into the process and a taxonomy with seven classes has been developed. These are (i) use of costs in the construction, (ii) post construction, (iii) GA methods, (iv) boosting, (v) bagging, (vi) multiple structures and (vii) stochastic methods. The algorithms have been described, the differences highlighted and a summary of observations made by the authors discussed. Additionally datasets used in all the different studies has been collated with the idea of guiding researchers to determine which of the existing algorithms may suit their purpose best.

In order to test the hypothesis and see if the aim can be achieved, a framework has been developed which builds on a specific Game Theory problem: the multi-armed bandit. This problem involves using exploration and exploitation techniques to solve it. Concepts from the multi-armed bandit game have been utilized in order to develop a new algorithm, viewing the pay-offs as a reduction in costs so that a compromise between decisions based on accuracy and decisions based on cost can be found. The multi-armed bandit game has been adapted to select the attributes in the decision tree induction. The result is an algorithm with many parameters which can alter behaviour in the hope that the correct behaviour can be found which suits each dataset and the costs belonging to that dataset.

Using 15 real-world datasets from the Machine Learning Repository, with varying values of test costs and misclassification costs, the algorithm has been investigated with regard to performance over the different parameter settings. Experiments were devised using single model induction version and a multiple-model induction version of the algorithm using all possible parameter settings for five varying values of lever pulls which are unique to each

dataset. For each dataset a range of misclassification costs were used which were higher than the test costs, lower than the test costs or a mixture of high and low values in relation to the test costs. The results of these experiments were compiled into a dataset which could then be analyzed and investigated. Each example is an outcome of a particular experiment, for a dataset and cost matrix and combination of parameter settings and has been classified into one of nine classes which reflect the cost and accuracy rate achieved by the experiment. The aim of the investigation was to summarize findings and decide (i) which parameters allow the algorithm to continue only when it is worthwhile to do so and what effect this has on cost and accuracy, (ii) which investigates the parameter settings which indicate how many lever pulls is better, which version of the algorithm is better and which strategy is better for a dataset given the test costs and misclassification costs, (iii) which investigates how choosing the best parameter settings and strategy for a dataset and cost matrix as an individual can improve on results and (iv) to determine whether guidelines to parameter settings can be found.

The main findings from the investigation in Chapter 5 were that:

- Of the four pre-pruning options, testing whether it is worthwhile to continue or not, the combination which tests the chosen attribute only, produces the majority of better low cost results
- Using the proportional stopping option along with class proportional stopping does not have as much impact on tree build as testing to see whether it is worthwhile to continue or not
- To produce the best accuracy, allowing the tree to be fully grown without forcing it to be stopped is the best option
- A higher value of lever pulls proved the better option in the majority of situations

- The multiple-model version proved to be the better option over the majority of datasets
- Investigations regarding the depth to look-ahead have produced results which indicate that the datasets, where the depth was increased to 2, did not show any improvement. These same results were obtained when the depth was 1 for two datasets iris and diabetes and no improvement for car and glass. This could be dataset related.

When processing the algorithm, using some combinations of parameters resulted in no model being built. It has been determined that to obtain higher accuracy rates it is necessary to grow a tree. Although this does result in a higher cost, the aim of the algorithm is to maximize accuracy and minimize cost and in order to do so a tree must be grown. The more exploration that takes place, the more likely it is to produce a model which will meet this aim. Better results are obtained for the majority of datasets by increasing the value of the parameter which allows more exploration; the number of lever pulls.

By treating each dataset/cost matrix combination as a separate entity, allows the flexibility of the algorithm to work to its full effect by choosing for each individual, the most appropriate strategy rather than choosing the same strategy for every cost matrix for a dataset. This is demonstrated in that choosing parameter settings which produced the lowest cost and using the corresponding accuracy for some cost matrices produces very good results but not for others, whilst choosing parameter settings which produced the highest accuracy and using the corresponding cost repeats this pattern. It is seldom that the same parameters produce the lowest cost and highest accuracy. By using the best strategy for an individual cost matrix, not the same strategy for every cost matrix, results in a flexible algorithm which gives the user more choice than with traditional cost-sensitive algorithms.

Guidelines have been produced, which attempt to predict the parameter settings for processing future datasets. Two approaches have been used in order to see if it is possible to set parameter settings as a guideline. One way was to examine all parameter settings producing good results and the other way was to process the results dataset using the accuracy-based algorithm J48. The outcome of these two approaches is that attributes which describe the dataset structure, that is class distribution or groups dictating a combination of attribute and values, need to be used to determine what parameter settings should be used.

In order to examine the performance of the MA_CSDT algorithm compared to well-known existing algorithms, an evaluation has been carried out using the 15 datasets with the misclassification costs and test costs which are also used in the investigation. The existing cost-sensitive algorithms are *EG2*; from the class costs used in the construction, *MetaCost*; from the class bagging, *AdaCost*; from the class boosting, *ICET*; from the GA methods class and *ACT*; from the stochastic approach class. Additionally the accuracy-based algorithm J48 was also used in order to provide a 'target' accuracy rate. The aim of the algorithm is to achieve the same accuracy rate as the accuracy-based algorithm, but do this at a lower cost than the other cost-sensitive algorithms. Each experiment and each combination of parameter settings has been examined in order to find the best option for each individual cost matrix for each dataset given two strategies; cost-based strategy which chooses the lowest cost obtained for a cost matrix and uses the corresponding accuracy rate and accuracy-based strategy which chooses the highest accuracy rate obtained for a cost matrix and uses the corresponding cost. These have then been compared to the cost and accuracy returned by each of the other algorithms.

The MA_CSDT algorithm can return the lowest cost for a cost matrix 68.2% of the time and return the highest accuracy for a cost matrix 34.98% of the time. Each time the highest accuracy is achieved, its corresponding cost is lower than that of J48. For six of the datasets, the algorithm MA_CSDT was able to achieve its aim of minimizing the costs whilst maximizing the accuracy rates. For an additional four datasets, the aim can be achieved by applying a strategy which is tailored to the individual dataset and its cost, thereby finding a compromise between accuracy-based decisions and cost-based decisions. For example, whilst it is desirable to obtain the highest accuracy rate as possible, this can be allowed to be reduced slightly, by one or two percent, in order to achieve a lower cost than existing cost-sensitive algorithms. By examining each cost matrix individually and by considering the accuracy rate obtained by the accuracy-based algorithm J48, in those datasets where the aim was not achieved by using one strategy for each cost matrix, a combination of both strategies were used thus achieving the aim. For some of these datasets, a higher accuracy rate than J48 had been achieved and in these cases by allowing a small sacrifice of the accuracy rate, a lower corresponding cost was found.

Using different strategies for individual cost matrices meant that the aim of the algorithm was achieved however there were two datasets which this technique did not help; those datasets where the information obtained from the test costs was insufficient; particularly those with low test costs; mushroom and soybean. In this case it is suggested adding extra information which can help decide whether a low test cost is hiding a bad split or a low test cost is associated with a good attribute would be a solution to this problem.

For the remaining three datasets; car, krk and nursery, MA_CSDT was unable to achieve its aim, producing only a low accuracy rate and being unable to minimize cost in all cases. It

should be noted however that on one of these datasets, krk, three of the existing cost-sensitive algorithms were also unable to achieve good results.

In conclusion, this thesis has developed a new algorithm and framework for cost-sensitive decision tree induction based on the principles of the multi-armed bandit problem. The algorithm has helped explore and confirm a research hypothesis, that cost-sensitive learning involves a trade-off between the decisions based on accuracy and decisions based on cost. By using a framework which explores strategies based on cost, a compromise between these viewpoints can be reached in the majority of cost-sensitive problems and that for those problems where the new algorithm is not as successful; a version containing extra information could be used in order that these problems can also be improved upon.

Future work

Although the algorithm can achieve its aim for some problems, there are some limitations at this time. The algorithm relies on the cost information available in order to determine the best attribute upon which to split the data. When this is limited for some reason, for example if the test costs are low or the misclassification costs are low, this presents the algorithm with a problem in that there is not enough information to determine which attribute has a low test cost hiding a bad split or a is a good split. Even though the algorithm is designed to look ahead, when all attributes have a low test cost it must rely on the misclassification costs only and in some cases particularly when these are also low, the algorithm can find itself unable to backtrack from a bad decision.

In order to overcome this limitation, it is suggested that a more complicated version of the multi-armed bandit algorithm can be used, one that has, as an add-on, a statistical measure in

order to combat attributes with lower test costs which may have been chosen simply because of the low test cost.

As noted in the discussion of the evaluation, the algorithm *EG2* was able to achieve lower costs for some of the cost matrices. This algorithm does not include misclassification costs in its statistical measure used to find the attribute upon which to split the data. However it does use the test costs in such a way that the value returned indicates whether the split is 'worth' the cost or not. Although the procedure does not react to a worthless split, simply choosing the best of what is on offer, it does at least make the best choice it can and indirectly the accuracy of the tree could be determined in a limited way.

The measurement this algorithm uses, as presented in equation 3.1, may be of use by supplying additional information regarding the 'worth' of the attribute. In the first instance it is suggested that equation (3.1) can be incorporated into the cost calculation used in the adaptation of the Multi-Armed Bandit problem as the simplest way of including extra information.

The experiments using the krk dataset, which produced overly large trees resulting in poor accuracy rates and high costs, could then be repeated using the extended version of the algorithm which includes the statistical measure equation (3.1). Different ways of incorporating this measure can be tried out. If not successful, research into more complicated multi-armed bandit scenarios can be undertaken to find a suitable algorithm which can be adapted for use by MA_CSDT for an improved measurement in order to find an attribute to exploit.

In Chapter 5, it has been suggested that sometimes the costs are such that it may not be worthwhile to induce a cost-sensitive decision tree. In these cases it was suggested that other techniques could be used instead. It may be worth investigating when this state is likely to be reached. In the case of the krk dataset for example, the accuracy-based algorithm produced a good accuracy rate and those cost-sensitive algorithms which used the same accuracy-based algorithm as a weak learner managed to reduce cost with only a small amount of reduction in accuracy. Perhaps when costs are too extreme pseudo-costs could be used therefore it would always be worthwhile in these cases. Other techniques could be used and compared to see if the answer lies in when not to use cost-sensitive trees but simply use the accuracy-based algorithm.

The result of the investigation and experimentation where the look-ahead depth was increased to 2 was inconclusive. It might be that the datasets selected for these experiments included datasets which would not require extensive look ahead searching. However to determine this required initial experimentation. It could also be the case that generating paths to a greater depth does not work for any dataset as this involves looking too many moves ahead; this has been noted in past literature that looking ahead too deeply can produce worse results (Murthy and Salzberg 1995). It is therefore suggested that using the information discovered in the investigation, one of the datasets where the results could be improved is chosen and used to investigate this further.

As discussed in Section 4.1, it was thought that the nature of the dataset could influence how the algorithm processed the dataset. It was noted that when examining the results from a previous comparison, there were inconsistencies. Even allowing for datasets which had the same test cost allocated to each of their attributes, inconsistent results were obtained. The

conclusion is that characteristics of the datasets, for example the number of attributes and how many different values these have, in addition to costs, have some influence on the ability of an algorithm to process the dataset. Although not examined in detail in this thesis, it is thought that an investigation using the results obtained in these experiments could be carried out in the first instance to determine if this conclusion is correct.

In conclusion, this thesis suggests the hypothesis that cost-sensitive learning is a trade-off between two alternative viewpoints, which are decisions based on accuracy and decisions based on cost. This thesis has presented an algorithm which tries to allow for different user and domain needs so that the user has more choice than a straightforward cost-sensitive algorithm which will simply aim to reduce costs which can have a detrimental effect on the accuracy rate, or as an alternative choose an accuracy-based algorithm which can have a detrimental effect on the cost. It is shown that allowing for the trade-off between decisions based on accuracy and those based on costs can achieve the aim of cost-sensitive learning which is to minimize costs whilst maximizing accuracy.

REFERENCES

ABE, N., ZADROZNY, B., LANGFORD, J. 2004. An iterative method for multi-class cost-sensitive learning. Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04, 3, August 22-25, Seattle, WA, USA, Kim W., Kohavi R., Gehrke J., DuMouchel W. (Eds).

AFIFI, A. A., CLARK, V. 1996. Computer-aided multivariate analysis. 3rd Edition, Chapman & Hall, London.

AUER, P. 2002. Using confidence bounds for exploitation-exploration trade-offs. Journal of Machine Learning Research, 3, 397-422.

AUER, P., CESA-BIANCHI, N., FREUD, Y., SCHAPIRE, R.E. 1995. Gambling in a rigged casino: the adversarial multi-armed bandit problem. Internal Report 223-98, DSI, Università di Milano, Italy, Foundations of Computer Science, Proceedings of 36th Annual Symposium, October 23-25, Milwaukee, USA, 322 – 331.

AUER, P., CESA-BIANCHI, N., FREUD, Y., SCHAPIRE, R.E. 2001. The non-stochastic multi-armed bandit problem. http://cseweb.ucsd.edu/~yfreund/papers/bandits.pdf, viewed on line 23 April 2013.

AUER, P., CESA-BIANCHI, N., FREUD, Y., SCHAPIRE, R.E. 2003. The non-stochastic multi-armed bandit problem. SIAM Journal on Computing, Vol. 32, Issue 1, 48 – 77.

AUER, P., CESA-BIANCHI, N., FISCHER, P. 2002. Finite-time analysis of the multi-armed bandit problem. Machine Learning 47, 235-256.

BASILI, V. AND WEISS, D. A. 1984. Methodology for Collecting Valid Software Engineering Data. IEEE Trans. Software Engineering. SE-10, 6, 728-738.

BAUER, E., KOHAVI, R. 1999. An empirical comparison of voting classification algorithms: bagging, boosting and variants. Machine Learning Vol. 36, Issue 1-2, 105-139.

BERRY, D. A., FRISTEDT, B. 1985. Bandit problems: sequential allocation of experiments. Monographs on Statistics and Applied Probability, London: Chapman & Hall, ISBN 0-412-24810-7.

BINMORE, K. 2007. Game theory a very short introduction. Oxford University Press Inc, New York.

BRADFORD, J. P., KUNZ, C., KOHAVI, R., BRUNK, C., BRODLEY, C. E. 1998a. Pruning decision trees with misclassification costs. 10th European Conference on Machine Learning (ECML-98), April 21-23 1998, Chemnitz, Germany, 131-136.

BRADFORD, J. P., KUNZ, C, KOHAVI R, BRUNK, C, BRODLEY, C. E. 1998b. Pruning decision trees with misclassification costs. Long version: online at http://robotics.stanford.edu/~ronnyk/prune-long.ps.gz (Accessed 8 April 2011).

BREIMAN, L., FRIEDMAN J. H., OLSEN R. A., STONE C. J. 1984. Classification and regression trees. Chapman and Hall/CRC, London.

CESA-BIANCHI, N., LUGOSI, G. 2006. Prediction, learning and games. Cambridge University Press, New York.

DAVIS, M. D., 1983. Game theory a nontechnical introduction. Dover Publications Inc, Mineola, NY, USA.

DAVIS, J. V., JUNGWOO, H., ROSSBACH, C. J. 2006. Cost-sensitive decision tree learning for forensic classification. In Proceedings of 17th European Conference on Machine Learning (ECML), LNCS 4212, Springer, 622-629.

DOMINGOS, P. 1999. MetaCost: A general method for making classifiers cost-sensitive. In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM New York, NY, USA, 155–164.

DONG, M., KOTHARI, R. 2001. Look-ahead based fuzzy decision tree induction. IEEE Transactions on Fuzzy Systems , Vol. 9, No 3, 461–468.

DORARD, L., SHAWE-TAYLOR, J. 2010. Gaussian process bandits for tree search. University College London, available on-line at http://ucl.academia.edu/LouisDorard/Papers/273653/Gaussian_Process_Bandits_for_Tree_Search, viewed 8/2/2011.

DORARD, L., GLOWACKA, D., SHAWE-TAYLOR, J. 2009. Gaussian process modelling of dependencies in multi-armed bandit problems. Proceedings of 10th International Symposium on Operational Research (SOR), September 23-25, Nova Gorica, Slovenia.

DRAPER, B. A., BRODLEY, C. E., UTGOFF, P. E. 1994. Goal-directed classification using linear machine decision trees. IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (9) September, 888 – 893.

ELKAN, C. 2001. The foundations of cost-sensitive learning. In Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI'01), Morgan Kaufmann, San Francisco, USA, Vol. 2, 973 – 978.

ESMEIR, S. AND MARKOVITCH, S. 2004. Lookahead-based algorithms for anytime induction of decision trees. Twenty-first international conference on Machine learning - ICML '04, 33, Brodley C.E. (Ed), 257-264.

ESMEIR, S. AND MARKOVITCH, S. 2007. Anytime induction of cost-sensitive trees. In Proceedings of The 21st Annual Conference on Neural Information Processing Systems (NIPS-2007), Vancouver, BC, Canada, 1-8.

ESMEIR, S. AND MARKOVITCH, S. 2008. Anytime induction of low-cost, low-error classifiers: a sampling-based approach. Journal of Artificial Intelligence Research 33, 1-31.

ESMEIR, S. AND MARKOVITCH, S. 2010. Anytime algorithms for learning resource-bounded classifiers. In Proceedings of the Budgeted Learning Workshop, ICML2010, Haifa, Israel, June 25.

ESMEIR, S. AND MARKOVITCH, S. 2011. Anytime learning of anycost classifiers. Machine Learning, Vol. 82, No 3, 445–473.

ESTRUCH, V., FERRI, C., HERNÁNDEZ-ORALLO, J., RAMÍREZ-QUINTANA, M. J. 2002. Re-designing cost-sensitive decision tree learning. In Workshop de mineria de datos y Aprendizaje, Iberamia, Seville, November 2002, 33–42.

FAN, W., STOLFO, S. J., ZHANG, J. CHAN, P. K. 1999. AdaCost: misclassification cost-sensitive boosting. 16th International Conference on Machine Learning, June 27-30 1999, Bled, Slovenia, 97-105.

FERRI, C., FLACH, P., HERNÁNDEZ-ORALLO, J. 2002. Learning decision trees using the area under the ROC curve. In 19th Machine Learning International workshop then conference, University of New South Wales, Sydney, Australia, 139–146.

FERRI-RAMÍREZ, C., HERNÁNDEZ, J., RAMIREZ, M. J. 2002. Induction of decision multi-trees using levin search. In International Conference on Computational Science, ICCS02, April 21 – 24, Amsterdam, The Netherlands, LNCS, Vol. 2329, 166-175.

FRANK, E., WITTEN, I. 1998. Reduced-error pruning with significance tests. Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.2272 (Accessed 8 April 2011).

FREAN, M. 1990. Small nets and short paths: optimizing neural computation. Doctoral thesis, Centre for Cognitive Science, University of Edinburgh.

FREITAS, A., COSTA-PEREIRA, A., BRAZDIL, P. 2007. Cost-sensitive decision trees applied to medical data. DaWak, September 3-7, Regensburg, Germany, LNCS 4654, 303-312.

FREUND, Y., SCHAPIRE, R.E. 1996. Experiments with a new boosting algorithm. 13th International Machine Learning workshop then conference, July 3-6, Bari, Italy, 148-156.

FREUND, Y., SCHAPIRE, R.E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences, 55 (1), Academic Press, Orlando, Florida, USA, 119-139.

GITTINS, J. C. 1989. Multi-armed bandit allocation indices. Wiley-Interscience Series in Systems and Optimization, Chichester: John Wiley & Sons, Ltd, ISBN 0-471-92059-2.

GREFENSTETTE, J. 1990. GENESIS: GENEtic Search Implementation System. A user's guide to GENESIS. http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/genetic/ga/systems/genesis/0.html
http://www.genetic-programming.com/c2003genesisgrefenstette.txt

GREINER, R, GROVE, A.J. and ROTH, D. 2002. Learning cost-sensitive active classifiers, Artificial Intelligence, Vol. 139, No 2, 137-174.

GRÜNEWÄLDER, S., AUDIBERT, J-Y., OPPER, M., SHAWE-TAYLOR, J. 2010. Regret bounds for Gaussian process bandit problems. Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, Vol. 9 of JMLR, Chia Laguna Resort, Sardinia, Italy.

HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., WITTEN, I.H. 2009. The WEKA data mining software: an update. SIGKDD Explorations, Vol. 11, Issue 1.

HAN, J., KAMBER, M. 2001. Data Mining: Concepts and techniques. Academic Press, London, Morgan Kaufman, CA, USA, ISBN 1-55860-489-8.

HART, A. E. 1985. Experience in the use of an inductive system in knowledge engineering. In Research and development in expert systems. M. A. Bramer (Ed.), Cambridge University Press.

HUNT, E. B., MARIN, J., STONE, P. J. 1966. Experiments in induction. New York, Academic Press.

KING, R. D., FENG, C., SUTHERLAND, A. 1995. STATLOG: Comparison of classification algorithms on large real-world problems. Applied Artificial Intelligence, 01/1995, 9, 289 – 333.

KNOLL, U., NAKHAEIZADEH, G., TAUSEND, B. 1994. Cost-sensitive pruning of decision trees. Proceedings of the 8th European Conference on Machine Learning ECML-94. Berlin, Germany, Springer-Verlag, 383-386.

KRETOWSKI, M. AND GRZEŚ, M. 2007. Evolutionary induction of decision trees for misclassification cost minimization. In Proceedings of 8th International Conference on Adaptive and Natural Computing Algorithms, April 11-14, Warsaw, Poland, ICANNGA, Part 1, LNCS 4431, Springer.

LI, J., LI, X., YAO, X. 2005. Cost-sensitive classification with genetic programming. IEEE Congress on Evolutionary Computation, 2114-2121.

LIN, F. Y. AND MCCLEAN, S. 2000. The prediction of financial distress using a cost-sensitive approach and prior probabilities. Proceedings of 17th International Conference on Machine Learning, ICML-2000, June 29-July 2, Stanford University, California, USA.

LING, C. X., YANG, Q., WANG, J., ZHANG, S. 2004. Decision trees with minimal costs. ACM International Conference Proceeding Series 21st international conference on Machine learning, Banff, Alberta, Canada, Article No. 69, ISBN: 1-58113-828-5. ACM Press New York, NY, USA.

LING, C. X., SHENG, V. S., BRUCKHAUS, T., MADHAVJI, N. H. 2006a. Maximum profit mining and its application in software development. Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06, August 20-23, Philadelphia, USA, 929.

LING, C., SHENG, V., YANG, Q. 2006b. Test strategies for cost-sensitive decision trees. IEEE Transactions on Knowledge and Data Engineering, 18(8), 1055-1067.

LIU, X. 2007. A new cost-sensitive decision tree with missing values. Asian Journal of Information Technology, 6(11), 1083–1090.

LIVARI, J., HIRSCHHEIM, R. A., KLEIN, H. K. 1998. A Paradigmatic analysis contrasting information systems development approaches and methodologies. Information Systems Research 9 (2). June 1998, 164 – 193.

LOMAX, S. 2006. An empirical comparison of cost-sensitive decision tree induction algorithms. MSc Thesis. University of Salford.

LOMAX, S., VADERA, S. 2009. An empirical comparison of cost-sensitive decision tree algorithms. In Proceedings from the fourth Conference on Intelligent Management Systems in Operations, OR, IMSIO4, 7-8 July 2009, University of Salford, Prof K.A.H. Kobbacy, Prof S. Vadera (Eds), published by OR Society, 35-47.

LOMAX, S., VADERA, S. 2011. An empirical comparison of cost-sensitive decision tree induction algorithms. Expert Systems The Journal of Knowledge Engineering, July, Vol. 28, No 3, 227 – 268.

LOMAX, S., VADERA, S. 2013. A survey of cost-sensitive decision tree induction algorithms. ACM Computing Surveys, Vol. 45, No. 2, Article 16. http://doi.acm.org/10.1145/2431211.2431215

LOMAX, S., VADERA, S. SARAEE, M. 2012. A multi-armed bandit approach to cost-sensitive decision tree learning. In Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference, 10-13 December 2012, Brussels, Belgium, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6406437, 162 – 168.

LOZANO, A.C., ABE, N. 2008. Multi-class cost-sensitive boosting with p-norm loss functions. Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '08, August 24-24, Las Vegas, USA, 506.

MARGINEANTU, D., DIETTERICH, T. 2003. A wrapper method for cost-sensitive learning via stratification. Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.1102 (accessed 7 April 2011).

MARGINEANTU, D. 2001. Methods for cost-sensitive learning. Doctoral Thesis, Oregon State University.

MAYNARD SMITH, J. 1984. Evolution and the theory of games. Cambridge University Press, Cambridge, UK.

MEASE, D., WYNER, A. J., BUJA, A. 2007. Boosted classification trees and class probability/quantile estimation. Journal of Machine Learning Research, 8, 409-439.

MEIR, R. AND RÄTSCH, G. 2003. An introduction to boosting and leveraging. Advanced Lectures on Machine Learning, Mendelson, S., Smola, A. (Eds), Springer, 119-184.

MERLER, S., FURLANELLO, C., LARCHER, B., SBONER, A. 2003. Automatic model selection in cost-sensitive boosting. Information Fusion, 4(1), 3-10.

MICHALEWICZ, Z. 1996. Genetic algorithms + data structures = evolution programs. Third Edition. Springer.

MINGERS, J. 1989. An empirical comparison of selection measures for decision-tree induction. Machine Learning, 3, 319-342.

MORET, S., LANGFORD, W. MARGINEANTU, D. 2006. Learning to predict channel stability using bio geomorphic features. Ecological Modelling, 191(1), 47-57.

MORRISON, D. 1976. Multivariate statistical methods. 2nd Edition. McGraw-Hill, New York.

MURTHY, S., KASIF, S., SALZBERG, S. 1994. A System for induction of oblique decision trees. Journal of Artificial Intelligence Research, 2, 1-32.

MURTHY, S., SALZBERG, S. 1995. Lookahead and pathology in decision tree induction. Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, Canada, 1025-1033.

NASH, J. F. 1950a. Equilibrium points in N-person games. Proceedings of the National Academy of Sciences of the United States of America, 36, 48 – 49.

NASH, J. F. 1950b. Non-cooperative games. Doctoral dissertation, Princeton University, reprinted in The essential John Nash (Harold W Kuhn and Sylvia Nasar, Eds), 53 – 84. Princeton University Press 2002.

VON NEUMANN, J. 1951. Various techniques used in connection with random digits. Monte Carlo methods. National Bureau Standards, 12, 36-38.

VON NEUMANN, J., MORGENSTERN, O. 1953. The theory of games and economic behaviour. Third Edition, Princeton University Press, Princeton, USA.

NI, A., ZHANG, S., YANG, S., ZHU, X. 2005. Learning classification rules under multiple costs. Asian Journal of Information Technology 4, 1080-1085.

NILSSON, N. J. 1965. Learning machines. McGraw-Hill, New York.

NORTON, S. W. 1989 Generating better decision trees. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. IJCAI-89, August, Detroit, Michigan, USA, 800-805.

NÚNEZ. 1991. The use of background knowledge in decision tree induction. Machine Learning 6, Kluwer Academic Publishers, Boston, 231-250.

OMIELAN, A. 2005. Evaluation of a cost-sensitive genetic classifier. MPhil Thesis, University of Salford.

OSBORNE. M. J., 2004. An introduction to game theory. Oxford University Press, New York, USA.

PAZZANI, M., MERZ, C., MURPHY, P., ALI, K., HUME, T., BRUNK, C. 1994. Reducing misclassification costs. In Proceedings of the 11th International Conference on Machine Learning.New Brunswick, New Jersey, USA, 217–225, Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Reducing+misclassification+costs#0 (accessed 7 April 2011).

QIN, Z., ZHANG, S., ZHANG, C. 2004. Cost-sensitive decision trees with multiple cost scales. Proceedings of the 17th Australian Joint Conference on Artificial Intelligence, December 4-6th Cairns, LNAI 3339, Springer-Verlag, Berlin, G. I. Webb and X Yu (Eds), 380-390.

QUINLAN, J. R. 1979. Discovering rules by induction from large collections of examples. Expert Systems in the micro electronic age, D Michie (Ed.), Edinburgh University Press, 168-201.

QUINLAN, J. R. 1983. Learning efficient classification procedures and their application to chess endgames. Machine Learning: an artificial intelligence approach, Michalski, Garbonell and Mitchell (Eds.) Tioga Publishing Company, Palo Alto.

QUINLAN, J. R. 1986. Induction of decision trees. Machine Learning 1, Kluwer Academic Publishers, Boston, 81-106.

QUINLAN, J. R. 1987. Simplifying decision trees. International Journal of Man-Machine Studies, 27, 221-234.

QUINLAN, J. R. 1993. C4.5: Programs for machine learning. Morgan Kaufman, San Mateo, CA.

QUINLAN, J. R., COMPTON, P. J., HORN, K. A., LAZARUS, L. 1987. Inductive knowledge acquisition: a case study. Applications of expert systems, Chapter 9, J Ross Quinlan (Ed), Based on the proceedings of the second Australian Conference. Turing Institute Press with Addison-Wesley Publishing Co. ISBN 0-201-17449-9, 137-156.

RASMUSEN, E. 2001. Games and information an introduction to game theory. Third Edition, Blackwell Publishers Ltd, Oxford, UK.

RISSANEN, J. 1978. Modelling by shortest data description. Automatica, 14, 465-471.

ROBBINS, H. 1952. Some aspects of the sequential design of experiments. Bulletin American Mathematical Society, 55, 527–535.

SCHAPIRE, R. E., FREUND, Y., BARTLETT, P., LEE, W. S. 1997. Boosting the margin: a new explanation for the effectiveness of voting methods. In Proceedings of the 14th International Conference on Machine Learning, Nashville, Tennessee, USA, 322-330.

SCHAPIRE, R., SINGER, Y. 1998. improved boosting algorithms using confidence-rated predictions. In Proceedings of the Eleventh Annual Conference on Computational Learning Theory, July 24-26, Madison, Wisconsin, USA, 80-91.

SCHAPIRE, R. E., SINGER, Y. 1999. Improved boosting algorithms using confidence-rated predictions. Machine Learning 37, (3), 297-336.

SCHAPIRE, R. E. 1999. A brief introduction to boosting. In Proceedings of the 16th International Joint Conference on Artificial Intelligence, IJCAI99, July 31 – August 6, City Conference Centre, Stockholm, Sweden, Vol. 2, 1401- 1406.

SHANNON, C. E. 1948. The mathematical theory of communication. The Bell System Technical Journal, Vol. 27, 379-423.

SHAWE-TAYLOR, J. 2010. Multivariate bandits and their applications. *In Intelligent Information Processing, IIP,* Presentation Slides, 15 October, held at Lowry Centre, Salford.

SHENG, S., LING, C. 2005. Hybrid cost-sensitive decision tree. 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, October 3-7, Porto, Portugal, LNCS, 3721, 274-284.

SHENG, S., LING, C., YANG, Q. 2005. Simple test strategies for cost-sensitive decision trees. In 16th European Conference on Machine Learning, ECML 2005, October 3-7, Porto, Portugal, LNCS 3720, 365-376.

SHENG, V. S., LING, C. X., NI, A., ZHANG, S. 2006. Cost-sensitive test strategies. Proceedings of 21st National Conference of Artificial Intelligence, July 16-20, Boston, Massachusetts, USA, AAAI Press.

VAN SOLINGEN, R., BASILI, V., CALDIERA, G., ROMBACH, H. D. 2002. Goal Question Metric (GQM) Approach. Encyclopaedia of Software Engineering. John Wiley & Sons, Inc.

SWETS, J., DAWES, R., MONAHAN, J. 2000. Better decisions through science. Scientific American, October, 82-87.

TAN, M., SCHLIMMER, J. 1989. Cost-sensitive concept learning of sensor use in approach and recognition. Proceedings of the 6th International Workshop on Machine Learning. ML-89, Ithaca, New York, 392-395.

TAN, M., SCHLIMMER, J. 1990. CSL: a cost-sensitive learning system for sensing and grasping objects. IEEE International Conference on Robotics and Automation. Cincinnati, Ohio.

TAN, M. 1993. Cost-sensitive learning of classification knowledge and its applications in robotics. Machine Learning, 13, 7-33.

TING, K. 1998. Inducing cost-sensitive decision trees via instance weighting. Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery, Berlin: Springer Verlag, 139-147.

TING, K. AND ZHENG, Z. 1998a. Boosting cost-sensitive trees. Proceedings of 1st International Conference on Discovery Science, Springer-Verlag, London, LNCS, 1532, 244–255.

TING, K. M., ZHENG, Z. 1998b. Boosting trees for cost-sensitive classifications. In *Machine Learning: ECML-98 10th European Conference on Machine Learning*, Chemnitz, Germany. Springer, 190-195.

TING, K. 2000a. An empirical study of Metacost using boosting algorithms. Proceedings of the 11th European Conference on Machine Learning, Springer-Verlag, London, LNCS 1810, 413–425.

TING, K. 2000b. A comparative study of cost-sensitive boosting algorithms. In Proceedings of the 17th International Conference on Machine Learning, June 29 - July 2, Stanford University, San Francisco, USA, 983-990.

TING, K. M. 2002. An instance-weighting method to induce cost-sensitive decision trees. IEEE Transactions on Knowledge and Data Engineering, 14, (3), May/June, 659-665.

TURNEY, P.D. 1995. Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm. Journal of Artificial Intelligence Research, 2, 369-409.

TURNEY, P. D. 2000. Types of cost in inductive concept learning. Workshop on Cost-Sensitive Learning at the 17th International Conference on Machine Learning (WCSL at ICML-2000), Stanford University, California, 15-21.

VADERA, S., VENTURA, D. 2001. A comparison of cost-sensitive decision tree learning algorithms. *Second European Conference in Intelligent Management Systems in Operations,* 3– 4 July, University of Salford, Operational Research Society, Birmingham, UK, 79-86.

VADERA, S. 2005a. Inducing cost-sensitive non-linear decision trees. Technical report. Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Inducing+cost-sensitive+non-linear+decision+trees#0 (Accessed 8 April 2011).

VADERA, S. 2005b. Inducing safer oblique trees without costs. Expert Systems: The International Journal of Knowledge Engineering and Neural Networks, Vol. 22, No 4, 206-221.

VADERA, S. 2010. CSNL: A cost-sensitive non-linear decision tree algorithm. ACM Transactions on Knowledge Discovery from Data (TKDD), Vol. 4, Issue 2, May 2010, Article 6, 1-25.

VERMOREL, J., MOHRI, M. 2005, Multi-armed bandit algorithms and empirical evaluation. In 16th European Conference on Machine Learning (ECML), October 3-7, Porto, Portugal, Springer, Vol. 3720, 437-448.

WEISS, S. M., KULIKOWSKI, C. A. 1991. Computer systems that learn classification and prediction methods from statistics, neural nets, machine learning and expert systems. Morgan Kaufmann Publishers Inc, California, USA, ISBN 1-55860-065-5

WINSTON, P. H. 1993. Artificial intelligence. 3rd ed. Addison Wesley, USA, ISBN 0-201-53377-4.

ZADROZNY, B., LANGFORD, J., ABE, N. 2003a. A simple method for cost-sensitive learning. Technical Report RC22666, IBM,2003. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.7.7947

ZADROZNY, B., LANGFORD, J., ABE, N. 2003b. Cost-sensitive learning by cost-proportionate example weighting. Third IEEE International Conference on Data Mining, November 19 - 22, 2003, Melbourne, Florida, USA, 435.

ZHANG, S., QIN, Z., LING, C., SHENG, S. 2005. "Missing is useful": missing values in cost-sensitive decision trees. IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No 12, 1689-1693.

ZHANG, S., ZHU, X., ZHANG, J., ZHANG, C. 2007. Cost-time sensitive decision tree with missing values. Knowledge Science, Engineering and Management 4798, 447-459.

ZHANG, S. 2010. Cost-sensitive classification with respect to waiting cost. Knowledge.-Based Systems, Vol. 23, No 5, 369-378.

APPENDIX

A1 Analysis of datasets used by studies from the survey

Table A1, given overleaf, shows the top 20 datasets used by the studies in Chapter 3. These datasets have been divided into groups; two class datasets, multi-class datasets and those which have been used as both two class and multi-class datasets. The table gives details of how many datasets each study used, the average number of datasets used and how many of the datasets are in the top 20. All datasets in the top 20 are available from the Machine Learning Repository[12]. Some of the studies have used private datasets or those from other sources.

The table also indicates whether the test costs and misclassification costs are provided. Other datasets which are not in the top 20 listing but are still useful in order to measure performance include: (i) the Soybean dataset which has 19 classes, (ii) Thyroid (NN), used by Turney (1995) and others, and is a larger version of the hypothyroid dataset and has 3 classes and (iii) Statlog Shuttle, a large dataset with 58,000 examples, useful to examine how an algorithm performs with a larger number of examples.

[12] http://archive.ics.uci.edu/ml/index.html

DATASETS			2-Class (from 52)													Both (2)		Multi-class (from 44)					
	Comments	How many out of top 20/Total number of datasets used in experiments	Breast-Wisconsin	Chess (krkp)	Credit Approval	Credit Approval (australia version) statlog	Echocardiogram	Hepatitis ‡	Horse Colic	Congressional Voting Records	Hypothyroid ‡	Mushroom	Pima Indian Diabetes ‡	Tic-tac-toe	Bupa Liver Disorders ‡	Heart Disease (cleveland) ‡	Annealing	Glass	Iris	Solar flare	Splice	Wine
Maximum no of datasets used in a study 28																						
Average no of datasets used in a study 8																						
Number of different studies using the dataset			14	5	5	7	5	12	5	6	5	8	16	10	11	18	6	8	5	7	6	6
ALGORITHM (52)	Studies 39																					
Use of costs during construction			*8*	*2*	*2*	*5*	*1*	*4*	*2*	*1*	*1*	*7*	*7*	*4*	*2*	*9*	*2*	*3*	*1*	-	*2*	-
GINI Altered Priors	Private(1)	3/4										x	x			x						
CS-ID3 *		0/1																				
IDX *		1/1								x												
EG2 *	Private(1)	0/3																				
LMDT	Private(1)	0/1																				
Cost-Minimization	Private(1)	3/4										x	x			x						
C4.5CS		11/22	x		x		x	x	x		x		x			x	x	x			x	
EvalCount, MaxCost, AvgCost		4/6															x	x	x		x	
DT with MC †		3/5	x			x										x						
DT with MC under RC †		3/5	x			x										x						
DTNB †		6/11	x	x		x						x		x		x						
CSNL	Private(1)	8/17	x		x			x	x				x		x	x		x				
LDT	Private(1)	4/6	x					x					x			x						
Performance †		1/1										x										
CSGain, CSGainRatio *		5/7	x			x		x					x		x							
CSTree	Private(1)	0/1																				
LazyTree †		6/10	x	x		x						x		x		x						
PM †		2/2										x		x								
CTS-DT †		2/2										x		x								
CS-C4.5 *	several used only	1/1											x									
Post construction			*1*	*1*				*1*		*1*	*1*	*2*	*2*	*1*	*1*	*1*	-	-	-	-	-	-
I-gain (Cost-Laplace prob)	Private(1)	3/4										x	x			x						
AUCSplit		9/25	x	x				x		x	x	x	x	x	x							
GA methods			*1*	-	-	*1*	-	*2*	-	-	-	-	*3*	-	*3*	*4*	-	*1*	-	-	-	*1*
ICET †		4/5						x					x		x	x						
ECCO †		4/4						x					x		x	x						
CGP	Private(1)	1/3														x						
GCT_MC		7/13	x			x							x		x	x		x				x
Boosting			*3*	*1*	*3*	-	*2*	*2*	*2*	*1*	*3*	-	*2*	*1*	*1*	*2*	*3*	*1*	-	*3*	*3*	-
UBoost, Cost-UBoost		11/22	x		x		x	x	x		x		x			x	x	x			x	
AdaCost	Private(1)	3/7	x		x						x											
CSB1, CSB2		13/14	x	x	x		x	x	x	x	x		x	x	x	x				x		
SSTBoost	Private(1)	0/1																				
GBSE, GBSE-T	KKD99•	3/6															x			x	x	
JOUS-Boost		0/6																				
Lp-CSB, Lp-CSB-PA, Lp-CSB-A	KDD99•	3/9															x			x	x	
Bagging			*2*	*1*	*1*	*1*	*2*	*2*	*2*	*2*	*1*	-	*2*	*1*	*2*	*2*	*2*	*2*	*1*	*2*	*2*	*2*
MetaCost		15/28	x		x		x	x	x	x			x		x	x	x	x	x	x	x	x
MetaCost_A, MetaCost_CSB		17/24	x	x		x	x	x	x	x	x		x	x	x	x	x	x		x	x	x
Cost plus Prior Probability, Cost-Only	Private(1)	0/1																				
Costing (Black box)	KDD98•/ DMEF••	0/2																				
B-PET, B-LOT	Private(1)	0/1																				
Multiple Structure			-	-	-	-	-	-	-	*1*	-	-	-	*1*	-	-	-	-	*1*	-	-	*1*
Multi-tree		4/15								x				x					x			x
Stochastic approaches			-	-	-	-	-	*2*	-	-	-	-	*2*	*2*	*2*	*2*	-	*2*	*2*	*2*	-	*2*
ACT †		9/25						x					x	x	x	x		x	x	x		x
TATA †		9/25						x					x	x	x	x		x	x	x		x

* incorporates test costs

† incorporates test costs + misclassification costs otherwise misclassification costs only

‡ indicates that test costs are available for this dataset, 5 in total

• indicates that misclassification costs are available for this dataset, 5 in total

•• indicates that part misclassification costs are available for this dataset, 1 in total

All top 20 datasets are available from the Machine Learning Repository

The total number of different datasets used is 98

- number of datasets from Machine Learning Repository used 76
- number of datasets from other sources used 22
- artificial (2), KDD repository (2), other researchers (2), DMEF (1), private: supplied from specific domains (10), not specified (5)

A2 Details of the datasets used in the main evaluation

A2.1 Annealing

Area: Physical

Steel annealing from the Machine Learning Repository. It is multi-class, imbalanced, with high test costs and high potential unique bandit paths. Costs obtained on consultation with domain expert. Missing values were indicated as – but are actually 'not applicable' and have not been removed. Attributes have been removed if all examples had only one value or less than 10 examples for other values. All continuous attributes have been discretized using WEKA.

attributes	no of values	test costs	discount	group
family	3	50.0	50.0	x
steel	8	50.0	50.0	x
carbon	10	50.0	50.0	x
hardness	7	50.0	50.0	x
tempRolling	2	50.0	50.0	x
condition	3	50.0	50.0	x
nonAgeing	2	50.0	50.0	x
surfQual	5	50.0	50.0	x
lustre	2	50.0	50.0	x
shape	2	50.0	50.0	x
oil	3	50.0	50.0	x
bore	3	50.0	50.0	x
thick	9	250.0	250.0	x
width	4	250.0	250.0	x
len	2	250.0	250.0	x
bf*	2	1500.0	1500.0	a
bt*	2	1500.0	1500.0	a
bwme*	3	1500.0	1500.0	a
bl*	2	1500.0	1500.0	a
chrom‡	2	1500.0	250.0	a
cbond‡	2	1500.0	250.0	a
ferro‡	2	1500.0	250.0	a
formability	6	2000.0	2000.0	x
strength	8	2000.0	2000.0	x
Total test costs		15850.0		

*These tests are individual tests. Each one is independent and cannot provide answers to the other attributes. Each one of these can provide answers to all attributes marked ‡.

‡values are obtained for these attributes by doing one of the tests marked *. So if an attribute from this group is chosen, one of the tests which get the values marked * must be carried out so the test cost is full price. If one of the tests either * or ‡ has been carried out before, attributes marked ‡ will have a value easily obtainable and discount will apply.

Dr K J Abrams, Technical Consultant, University of Salford has provided an insight into this dataset along with advice regarding the allocation of the test costs and the groupings of the attributes.

Details of the class distribution of the annealing dataset are as follows:

	Frequency	Percent	Cumulative Percent
1	8	.9	.9
2	99	11.0	11.9
3	684	76.2	88.1
5	67	7.5	95.5
U	40	4.5	100.0
Total	898	100.0	

A2.2 Breast Cancer

Area: Medical

This version was obtained from Esmeir and Markovitch (2008), originally from the Machine Learning Repository; the costs were allocated by Esmeir and Markovitch (2008) also. It is one of three domains originally provided by Oncology Institute. Nine examples which had missing values have been removed by Esmeir and Markovitch.

attributes	no of values	test cost	discount	group
breast	2	5.02877	5.02877	x
menopause	3	9.2464	1.84928	a
breastquad	5	18.4302	18.4302	x
age	9	24.2124	16.8152	a
irradiat	2	40.8691	40.8691	x
tumorsize	12	41.8689	41.8689	x
nodecaps	2	61.1264	53.7293	a
degmalig	3	75.1722	75.1722	x
invnodes	13	93.0188	93.0188	x
Total test costs		368.9732		

It is a two class dataset; no recurrence events and recurrence events and is imbalanced. The class distribution is as follows:

	Frequency	Percent	Cumulative Percent
no recurrence events	196	70.8	70.8
recurrence events	81	29.2	100.0
Total	277	100.0	

A2.3 Car Evaluation

Area: Social

Originally from the Machine Learning Repository, this file is the one used by Esmeir and Markovitch (2008), along with the costs. It evaluates cars based on buying price, maintenance, overall price, technical characteristics and comfort and capacity. Structural information has been removed from the dataset leaving the attributes shown in the table. There were no missing values.

attributes	no of values	test costs	discount	groups
maint	4	5.58562	5.58562	x
lugboot	3	6.84251	6.84251	x
doors	4	20.6264	20.6264	x
buying	4	35.4139	7.08277	A
persons	3	91.6271	63.296	A
safety	3	98.6441	70.313	A
Total test costs		258.7396		

This dataset is a multi-class dataset with an imbalanced distribution.

	Frequency	Percent	Cumulative Percent
acc	384	22.2	22.2
good	69	4.0	26.2
unacc	1210	70.0	96.2
vgood	65	3.8	100.0
Total	1728	100.0	

A2.4 Pima Indian Diabetes

Area: Medical

Originally from the Machine Learning Repository but this version is from Esmeir and Markovitch (2008), with costs from Turney (1995). The classes are whether an example shows signs of diabetes, with class value 1 indicating tested positive. The population lives near Phoenix, Arizona. All attributes which were continuous were discretized using WEKA.

attributes	no of values	test costs	discount	groups
timespreg	2	1.00	1.00	x
massindex	2	1.00	1.00	x
pedigree	2	1.00	1.00	x
age	2	1.00	1.00	x
glucosetol	4	17.61	15.51	A
insulin	3	22.78	20.68	A
Total test costs		44.39		

The class distribution, which has a 65/35 distribution, is as follows:

	Frequency	Percent	Cumulative Percent
0	500	65.1	65.1
1	268	34.9	100.0
Total	768	100.0	

A2.5 Solar Flare

Area: Physical

The original dataset contains three potential classes, one for the number of times a certain type of solar flare occurred in a 24 hour period. Each instance represents captured features for 1 active region of the sun. Esmeir and Markovitch (2008) have used flare1.data and the class being used is C-class flares. There are no missing values.

attributes	no of values	test costs	discount	group
codeforclass	6	4.36544	4.36544	x
codeforspotsize	6	6.60975	1.32195	B
codeforspotdist	4	7.69262	7.54576	B
activity	2	8.21288	8.21288	x
evolution	3	10.0288	2.00576	B
previous	3	39.8118	11.0989	B
complex	2	51.9166	26.275	B
region	2	69.6765	64.3887	B
area	2	87.3879	87.3879	x
soptarea	2	96.5457	96.5457	x
Total test costs		382.248		

There is an imbalanced class distribution.

	Frequency	Percent	Cumulative Percent
0	287	88.9	88.9
1	29	9.0	97.8
2	7	2.2	100.0
Total	323	100.0	

A2.6 Glass Identification

Area: Forensic Science

This dataset deals with the identification of types of glass left at crime scenes. Esmeir and Markovitch's file was used along with their costs. Attributes have been discretized by WEKA. There are no missing values.

attributes	no of values	test cost	discount	group
a6	4	13.3987	13.3987	x
a0	3	40.2861	32.9151	a
a1	2	49.5373	42.1663	a
a3	3	67.0591	59.6881	a
a7	2	70.2323	62.8613	a
a5	4	75.0775	75.0775	x
a2	2	78.2133	70.8423	a
Total test costs		393.8043		

The classes are listed as follows:

1. Building windows float
2. Building windows non-float
3. Vehicle windows float
4. Vehicle windows non float processed (not in dataset)

5. Containers
6. Tableware
7. Headlamps

This results in a 6 class dataset with the following class distribution:

	Frequency	Percent	Cumulative Percent
1	70	32.7	32.7
2	76	35.5	68.2
3	17	7.9	76.2
5	13	6.1	82.2
6	9	4.2	86.4
7	29	13.6	100.0
Total	214	100.0	

A2.7 Heart Cleveland

Area: Medical

This dataset is from the Machine Learning Repository, but the actual file used is the one prepared by Esmeir and Markovitch (2008). The costs used are those provided by Turney (1995) although the group data has been altered slightly as the ICET implementation used in the evaluation can only process one group datasets. There were some missing values but these have been removed by Esmeir and Markovitch. The dataset indicates the absence or presence of heart disease.

attributes	no of values	test costs	discount	group
age	2	1.0	1.0	x
sex	2	1.0	1.0	x
pain	4	1.0	1.0	x
fbs	2	5.2	5.2	x
restecg	3	15.5	15.5	x
exang	2	87.3	87.3	x
oldpeak	2	87.3	87.3	x
slope	3	87.3	87.3	x
fca	2	100.9	100.9	x
thal	3	102.9	1.0	b
thalach	2	102.9	1.0	b
Total test costs		592.3		

It is a two class dataset with an even class distribution:

	Frequency	Percent	Cumulative Percent
0	160	53.9	53.9
1	137	46.1	100.0
Total	297	100.0	

A2.8 Hepatitis

Area: Medical

Originally from the Machine Learning Repository, there are missing values, but this version was obtained from Esmeir and Markovitch (2008). There is no indication of what they did with the missing values; however there is one less example. The continuous attributes were discretized using WEKA, and the costs were those used by Turney (1995).

attributes	no of values	test costs	discount	group
sex	2	1.00	1.00	x
steroid	2	1.00	1.00	x
antiviral	2	1.00	1.00	x
fatigue	2	1.00	1.00	x
malaise	2	1.00	1.00	x
anorexia	2	1.00	1.00	x
liverbig	2	1.00	1.00	x
liverfirm	2	1.00	1.00	x
spleen	2	1.00	1.00	x
spiders	2	1.00	1.00	x
ascites	2	1.00	1.00	x
varices	2	1.00	1.00	x
histology	2	1.00	1.00	x
bilirubin	2	7.27	5.17	A
albumin	3	7.27	5.17	A
protime	3	8.30	6.20	A
Total test costs		35.84		

The class distribution is unbalanced as follows:

	Frequency	Percent	Cumulative Percent
1	32	20.8	20.8
2	122	79.2	100.0
Total	154	100.0	

A2.9 Iris Plants

Area: Botanical

This dataset contains 3 classes of iris plant. The aim is to predict the type of plant. There are no missing values and the attributes have been discretized by WEKA. This version is from Esmeir and Markovitch (2008) along with costs.

attributes	no of values	test costs	discount	groups
sl	3	7.65056	7.65056	x
sw	3	22.5831	22.5831	x
pl	3	78.2133	78.2133	x
pw	3	98.2458	98.2458	x
Total test costs		206.6928		

The class distribution is even:

	Frequency	Percent	Cumulative Percent
Iris-setosa	50	33.3	33.3
Iris-versicolor	50	33.3	66.7
Iris-virginica	50	33.3	100.0
Total	150	100.0	

A2.10 Chess Endgame White King and Rook against Black King (krk)

Area: Game

This dataset is a stored game with theoretic values for the enumerated elements. It denotes whether positions are won for either side or include the depth of win (number of moves), with Black to move, positions drawn or lost in N moves. The class is the optimal depth of win for white in 0 – 16 moves otherwise drawn.

This dataset is available from the Machine Learning Repository but this version was obtained, along with costs, from Esmeir and Markovitch (2008).

attribute	no of values	test costs	discount	group
wrr	8	7.11229	7.11229	x
wrf	8	7.3094	7.3094	x
wkf	4	7.57644	7.57644	x
bkr	8	70.0909	70.0909	x
bkf	8	78.6536	78.6536	x
wkr	4	89.3119	89.3119	x
Total test costs		260.0522		

The class distribution is as follows:

	Frequency	Percent	Cumulative Percent
0	27	.1	.1
1	78	.3	.4
2	246	.9	1.3
3	81	.3	1.5
4	198	.7	2.2
5	471	1.7	3.9
6	592	2.1	6.0
7	683	2.4	8.5
8	1433	5.1	13.6
9	1712	6.1	19.7
10	1985	7.1	26.8
11	2854	10.2	36.9
12	3597	12.8	49.7
13	4194	14.9	64.7
14	4553	16.2	80.9
15	2166	7.7	88.6
16	390	1.4	90.0
d	2796	10.0	100.0
Total	28056	100.0	

A2.11 Mushroom

Area: Botanical

This dataset is from the Machine Learning Repository with costs used by Lomax and Vadera (2011). There is a high potential unique bandit paths value with low test costs. It is a two class dataset with an even distribution. There are only nominal values. Missing values all in attribute stalk-root, coded as value x and left in the dataset. Attribute veil-type was removed

as it only has one value. The aim of the dataset is to classify edible and poisonous mushrooms.

attributes	no of values	test costs	discount	group
bruises	2	1.0	1.0	x
odor	9	1.0	1.0	x
pop	6	1.0	1.0	x
hab	7	1.0	1.0	x
ss	2	2.0	2.0	x
sr	5	2.0	2.0	x
ssar	4	2.0	2.0	x
ssbr	4	2.0	2.0	x
scar	9	2.0	2.0	x
scbr	9	2.0	2.0	x
ga	2	3.0	3.0	x
gs	2	3.0	3.0	x
gsize	2	3.0	3.0	x
gc	12	3.0	3.0	x
cs	6	4.0	4.0	x
csur	5	4.0	4.0	x
cc	10	4.0	4.0	x
rn	3	5.0	5.0	x
rt	5	5.0	5.0	x
spc	9	6.0	6.0	x
vc	4	7.0	7.0	x
Total test costs		63.0		

	Frequency	Percent	Cumulative Percent
0	4208	51.8	51.8
1	3916	48.2	100.0
Total	8124	100.0	

A2.12 Nursery

Area: Social

Evaluation of applications for nursery schools, with the final decisions resting on occupation of parents, family structure, financial standing, social and health. This dataset is available from the Machine Learning Repository but this version is from Esmeir and Markovitch

(2008) along with costs. Their version has less examples than the Machine Learning Repository one (which was 12960). However there is no explanation regarding their processing of the dataset.

attributes	no of values	test costs	discount	groups
finance	2	8.21288	1.64258	A
children	4	9.86595	3.29564	A
housing	3	10.004	1.0	A
parents	3	11.0199	4.44957	A
hn	5	11.4081	11.4081	x
social	3	15.8234	9.25308	A
form	4	20.949	20.949	x
health	3	98.6441	98.6441	x
Total test costs		185.9273		

The class distribution (apart from 2 classes with very few examples) is balanced, as follows:

	Frequency	Percent	Cumulative Percent
not recom	2900	33.3	33.3
priority	3644	41.9	75.2
recommend	2	.0	75.2
spec prior	1829	21.0	96.2
very recom	328	3.8	100.0
Total	8703	100.0	

A2.13 Soybean

Area: Botanical

This dataset diagnoses the diseases of the soybean plant. It is from the Machine Learning Repository, with the costs as used by Lomax and Vadera (2011). Only 15 of the classes are traditionally used. All examples with missing values have been removed. Most of them have multiple values missing:

attributes	no of values	test costs	discount	group
plant stand	2	5.00	5.00	x
hail	2	5.00	5.00	x
plant growth	2	5.00	5.00	x
leaves	2	5.00	5.00	x
leafshread	2	5.00	5.00	x
leafmalf	2	5.00	5.00	x
stem	2	5.00	5.00	x
lodging	2	5.00	5.00	x
fruiting bodies	2	5.00	5.00	x
mycelium	2	5.00	5.00	x
sclerotia	2	5.00	5.00	x
seed	2	5.00	5.00	x
mold growth	2	5.00	5.00	x
seed discolour	2	5.00	5.00	x
seed size	2	5.00	5.00	x
shrivelling	2	5.00	5.00	x
date	7	10.00	2.0	a
precip	3	10.00	2.0	a
temp	3	10.00	2.0	a
crop hist	4	10.00	2.0	a
area damaged	4	10.00	2.0	a
severity	3	10.00	2.0	a
seed tmt	3	10.00	2.0	a
germination	3	10.00	2.0	a
leafspotshalo	3	10.00	2.0	a
leafspotsmarg	3	10.00	2.0	a
leafspotsize	3	10.00	2.0	a
leaf mild	3	10.00	2.0	a
stem cankers	4	10.00	2.0	a
canker lesion	4	10.00	2.0	a
external decay	2	10.00	2.0	a
int discolour	3	10.00	2.0	a
fruit pods	3	10.00	2.0	a
fruit spots	4	10.00	2.0	a
roots	3	10.00	2.0	a
Total test costs		270.0		

- external decay has no value 2 – watery
- fruitpods now has no value 2 – few present
- fruitspots has no value 3 – distort (did not have one anyway).

This makes no overall difference to the class distribution, which is fairly evenly spread over the 15 classes:

	Frequency	Percent	Cumulative Percent
diaporthe-stem-canker	20	3.6	3.6
charcoal-rot	20	3.6	7.1
rhizoctonia-root-rot	20	3.6	10.7
phytophthora-rot	20	3.6	14.2
brown-stem-rot	44	7.8	22.1
powdery-mildew	20	3.6	25.6
downy-mildew	20	3.6	29.2
brown-spot	92	16.4	45.6
bacterial-blight	20	3.6	49.1
bacterial-pustule	20	3.6	52.7
purple-seed-stain	20	3.6	56.2
anthracnose	44	7.8	64.1
phyllosticta-leaf-spot	20	3.6	67.6
alternarialeaf-spot	91	16.2	83.8
frog-eye-leaf-spot	91	16.2	100.0
Total	562	100.0	

A2.14 Tic-tac-toe Endgame

Area: Game

This encodes the complete set of possible board configurations at the end of tic-tac-toe where x is assumed to have played first. Target concept is ‘win for x’ (true when x has one of 8 possible ways to create 3 in a row). The attributes correspond to the squares. Costs used are those in Lomax and Vadera (2011).

attributes	no of values	test costs	discount	groups
a0	3	1.0	1.0	x
a1	3	1.0	1.0	x
a2	3	1.0	1.0	x
a10	3	1.0	1.0	x
a11	3	1.0	1.0	x
a12	3	1.0	1.0	x
a20	3	1.0	1.0	x
a21	3	1.0	1.0	x
a22	3	1.0	1.0	x
Total test costs		9.0		

It is a 2 class dataset with a 65/35 distribution:

	Frequency	Percent	Cumulative Percent
negative	332	34.7	34.7
positive	626	65.3	100.0
Total	958	100.0	

A2.15 Wine Recognition

Area: Physical

The dataset comprises of the results of chemical analysis of wines grown in the same region in Italy, but derived from 3 different cultivars. Analysis determines the quantities of 13 constituents found in each of the 3 types of wine.

This is available from the Machine Learning Repository, but this file is from Esmeir and Markovitch (2008) along with their costs. All attributes are continuous and so was discretized by WEKA.

attributes	no of values	test costs	discount	groups
a3	2	5.86631	5.86631	x
a8	2	14.6291	14.6291	x
a4	2	30.1854	6.03708	b
a6	3	39.8244	39.8244	x
a1	3	42.3829	18.2345	b
a5	2	47.2174	47.2174	x
a2	3	48.6759	48.6759	x
a9	1	52.5906	52.5906	x
a12	3	73.7248	49.5765	b
a10	3	74.856	74.856	x
a11	4	83.4669	83.4669	x
a13	4	84.78	84.78	x
a7	4	98.5054	98.5054	x
Total test costs		697.0051		

This is a 3 class dataset with an even distribution, as shown:

	Frequency	Percent	Cumulative Percent
1	59	33.1	33.1
2	71	39.9	73.0
3	48	27.0	100.0
Total	178	100.0	

A3 Details of the misclassification costs used in all experiments

A3.1 2-class datasets

cost matrix	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
class															
1	1	1	1	1	1	1	1	1	10	50	100	500	1000	5000	10000
2	10000	5000	1000	500	100	50	10	1	1	1	1	1	1	1	1

A3.2 3-class datasets

cost matrix	1	2	3	4	5	6	7	8	9
class									
1	1	100	10	1	10	5	150	250	200
2	10	1	100	5	1	10	200	150	250
3	100	10	1	10	5	1	250	200	150

A3.3 4-class datasets

cost matrix	1	2	3	4	5	6	7	8	9	10	11	12
class												
1	1	500	100	10	1	20	10	5	150	300	250	200
2	10	1	500	100	5	1	20	10	200	150	300	250
3	100	10	1	500	10	5	1	20	250	200	150	300
4	500	100	10	1	20	10	5	1	300	250	200	150

A3.4 5-class dataset: annealing

cost matrix	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
class															
1	100	10000	5000	1000	500	150	350	300	250	200	1000	5000	4000	3000	2000
2	500	100	10000	5000	1000	200	150	350	300	250	2000	1000	5000	4000	3000
3	1000	500	100	10000	5000	250	200	150	350	300	3000	2000	1000	5000	4000
4	5000	1000	500	100	10000	300	250	200	150	350	4000	3000	2000	1000	5000
5	10000	5000	1000	500	100	350	300	250	200	150	5000	4000	3000	2000	1000

A3.5 5-class dataset: nursery

cost matrix	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
class															
1	1	1000	500	100	10	1	50	20	10	5	150	350	300	250	200
2	10	1	1000	500	100	5	1	50	20	10	200	150	350	300	250
3	100	10	1	1000	500	10	5	1	50	20	250	200	150	350	300
4	500	100	10	1	1000	20	10	5	1	50	300	250	200	150	350
5	1000	500	100	10	1	50	20	10	5	1	350	300	250	200	150

A3.6 6-class dataset: glass

cost matrix	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
class																		
1	1	1000	500	100	50	10	1	70	50	20	10	5	150	400	350	300	250	200
2	10	1	1000	500	100	50	5	1	70	50	20	10	200	150	400	350	300	250
3	50	10	1	1000	500	100	10	5	1	70	50	20	250	200	150	400	350	300
4	100	50	10	1	1000	500	20	10	5	1	70	50	300	250	200	150	400	350
5	500	100	50	10	1	1000	50	20	10	5	1	70	350	300	250	200	150	400
6	1000	500	100	50	10	1	70	50	20	10	5	1	400	350	300	250	200	150

A3.6 15-class dataset: soybean

cost matrix	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
class															
1	1	700	650	600	550	500	450	400	350	300	250	200	150	100	50
2	50	1	700	650	600	550	500	450	400	350	300	250	200	150	100
3	100	50	1	700	650	600	550	500	450	400	350	300	250	200	150
4	150	100	50	1	700	650	600	550	500	450	400	350	300	250	200
5	200	150	100	50	1	700	650	600	550	500	450	400	350	300	250
6	250	200	150	100	50	1	700	650	600	550	500	450	400	350	300
7	300	250	200	150	100	50	1	700	650	600	550	500	450	400	350
8	350	300	250	200	150	100	50	1	700	650	600	550	500	450	400
9	400	350	300	250	200	150	100	50	1	700	650	600	550	500	450
10	450	400	350	300	250	200	150	100	50	1	700	650	600	550	500
11	500	450	400	350	300	250	200	150	100	50	1	700	650	600	550
12	550	500	450	400	350	300	250	200	150	100	50	1	700	650	600
13	600	550	500	450	400	350	300	250	200	150	100	50	1	700	650
14	650	600	550	500	450	400	350	300	250	200	150	100	50	1	700
15	700	650	600	550	500	450	400	350	300	250	200	150	100	50	1

A3.7 18-class dataset: krk

cost matrix	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
class																		
1	1	850	800	750	700	650	600	550	500	450	400	350	300	250	200	150	100	50
2	50	1	850	800	750	700	650	600	550	500	450	400	350	300	250	200	150	100
3	100	50	1	850	800	750	700	650	600	550	500	450	400	350	300	250	200	150
4	150	100	50	1	850	800	750	700	650	600	550	500	450	400	350	300	250	200
5	200	150	100	50	1	850	800	750	700	650	600	550	500	450	400	350	300	250
6	250	200	150	100	50	1	850	800	750	700	650	600	550	500	450	400	350	300
7	300	250	200	150	100	50	1	850	800	750	700	650	600	550	500	450	400	350
8	350	300	250	200	150	100	50	1	850	800	750	700	650	600	550	500	450	400
9	400	350	300	250	200	150	100	50	1	850	800	750	700	650	600	550	500	450
10	450	400	350	300	250	200	150	100	50	1	850	800	750	700	650	600	550	500
11	500	450	400	350	300	250	200	150	100	50	1	850	800	750	700	650	600	550
12	550	500	450	400	350	300	250	200	150	100	50	1	850	800	750	700	650	600
13	600	550	500	450	400	350	300	250	200	150	100	50	1	850	800	750	700	650
14	650	600	550	500	450	400	350	300	250	200	150	100	50	1	850	800	750	700
15	700	650	600	550	500	450	400	350	300	250	200	150	100	50	1	850	800	750
16	750	700	650	600	550	500	450	400	350	300	250	200	150	100	50	1	850	800
17	800	750	700	650	600	550	500	450	400	350	300	250	200	150	100	50	1	850
18	850	800	750	700	650	600	550	500	450	400	350	300	250	200	150	100	50	1

A4 Summary of the attributes in the results dataset

attribute	description
id	numerical identifier
dataset	dataset name
ExpId	experiment identifier
attributes	number of attributes in dataset
maxValue	highest number of attribute values in a dataset
potPaths	potential unique bandit paths calculated for dataset and given depth
grpPotPaths	potential unique bandit paths values grouped
discretize	whether discretized in WEKA
distribution	class distribution of dataset
multiclass	whether multi-class dataset or not
mcost	misclassification cost identifier
groupMiscost	which group is the misclassification cost in
high_mc	what is the highest misclassification cost in the matrix
descMcost	group which identifies relationship of misclassification to test costs
tot_mcost	total of misclassification costs in the matrix
tot_tcost	total of test costs for the dataset
highest_tc	what is the highest test cost for the dataset
mcosttcost	ratio of highest misclassification cost/highest test cost
grpRatio	ratio value grouped
group	whether the test costs have groups or not
strat	what strategy has been used
model	how many models generated
label	which labeling system used
depth	what look-ahead depth used
oneVal	what strategy with attribute with one value to use
PId	what P category used (unique to dataset), as detailed in Table 11
P	what is the actual value of P used, as detailed in Table 11
worthPruneCombo	what pre-pruning combination used
prop	whether proportional stopping used
cprop	whether class proportional stopping used
dn_cost	'do nothing' cost to classify
dn_accuracy	'do_nothing' accuracy rate
cost	actual cost returned
accuracy	actual accuracy returned
trees	whether a tree was grown or not
costCat	category the cost value belongs to
accuracyCat	category the accuracy value belongs to
adjAccCat	category when allowing for n-class accuracy
bestCost	if this was the lowest cost for the matrix
bestAccuracy	if this was the highest accuracy for the matrix
altCost	if no model grown, next lowest cost with model
altAcc	if no model grown, next highest accuracy with model
costAccBest	if example is bestCost and bestAccuracy
altCostAccBest	if example is altCost and bestAccuracy
altCAcc	if example is altCost and altAcc
bestCostaltAccBest	if example is bestCost and altAcc
goodResults	combined class groups
class	class to which example has been assigned

A5 Parameter settings % frequencies of best and worst results all examples and trees only plus frequency of trees not grown or grown

Parameter Settings	best (examples)	best (trees)	worst (examples)	worst (trees)	NO TREE	TREE
strategy – if the lowest cost desired, if the highest accuracy desired, if single version none						
cost	32.2	35.2	1.13	2.0	55.4	44.5
accuracy	35.4	42.9	1.39	2.29	52	47.9
none	32.9	37.0	1.44	2.6	55	44.9
model - (indicates the number of models which will be generated, if single version 1						
1	32.9	37.0	1.44	2.6	55	44.9
10	34.1	39.8	1.28	2.2	53.5	46.4
20	22.2	37.5	0.27	0.46	40.8	59.1
label – if the hybrid labeling system is used HYBRID, if cost-sensitive labeling used COST						
HYBRID	37.8	40.2	1.44	2.2	48.5	51.4
COST	29.2	37.1	1.21	2.4	59.8	40.1
oneVal – indicates what do to if attribute chosen has only one value						
stop	33.0	37.4	1.27	2.2	53.8	46.1
ignore	33.8	39.1	1.37	2.4	54	45.9
nextBest	34.2	40.1	1.35	2.3	54	45.9
P lever Pulls – using the potential unique bandit calculation as guideline, id 3 is actual value						
1	33.3	38.5	1.29	2.2	53.8	46.2
2	33.8	38.9	1.33	2.3	54.2	45.8
3	33.9	39.2	1.33	2.3	54.3	45.7
4	34.0	39.2	1.31	2.2	54.4	21.5
5	32.9	38.0	1.45	2.5	53	47
6	78.3	78.3	0	0	0	100
7	76.6	76.6	0	0	0	100
-pre-pruning option – implementing Ling et al. (2004) method used before start of process / used when the attribute has been chosen; a combination of these two methods						
a: true/true	29.2	29.6	0.48	0	74.6	25.3
b: true/false	29.9	26.8	0.46	1.2	68.7	31.2
c: false/true	32.3	44.7	0.48	0.009	72.5	27.4
d: false/false	43.4	43.4	3.9	3.9	0	100
-prop – if no. of examples below 6% of original proportion of training dataset, or value as stated						
FALSE	33.6	38.3	1.8	3.5	54	45.9
TRUE	34.5	40.3	0.65	0.92	53.9	46
0.15	0	0	8.1	10.2	54.8	45.1
0.2	0	0	7.6	8.9	54.9	45
-cprop – if no of examples in majority class is 90% of examples at node, or value as stated						
FALSE	33.6	38.3	1.87	3.5	54	45.9
TRUE	34.5	40.3	0.65	0.92	53.9	46
0.8	0	0	7.87	9.6	54.9	45

APPENDIX D

The following documents are on the attached DVD and are named as indicated.

D1 Do nothing accuracy versus accuracy obtained by a number of classes

D2 Graphs showing datasets with misclassification costs reduced into groups

D3 Best results obtained for each dataset

D4 Comparing mean values obtained with values obtained by a given strategy

D5 Annotated graphs showing datasets processed using pruned versions of the cost-sensitive algorithms

D6 Annotated graphs showing datasets processed using un-pruned versions of the cost-sensitive algorithms

D7 Tables showing datasets processed using pruned versions of the cost-sensitive algorithms

D8 Tables showing datasets processed using un-pruned versions of the cost-sensitive algorithms

D9 AnalysisOfResultsFile.csv which contains the results from all the experiments