DEVELOPMENT OF NEW COST-SENSITIVE BAYESIAN NETWORK LEARNING ALGORITHMS

Eman Bashir Nashnush

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

Submitted in Fulfilment of the Requirements of the Degree of Doctor of Philosophy, November 2015

TABLE of CONTENTS

LIST of FIGURES

LIST of TABLES

LIST of ABBREVIATIONS and SYMBOLS

AI	Artificial Intelligence
BD	Bayesian Dirichlet
BDe	Bayesian Dirichlet likelihood-equivalence
BDeu	Bayesian Dirichlet likelihood-equivalence uniform joint distribution
BNs	Bayesian Networks
CI	Conditional Probability Tables
CL tree	Chow-Liu tree
CLL	Conditional log likelihood
CSC	CostSensitiveClassifier
CS-BN	Cost-sensitive Bayesian networks
DT	Decision tree
ECCO	Evolutionary Classifier with Cost Optimization
FN	False Negative
FP	False Positive
GAs	Genetic Algorithms
ICET	Inexpensive Classification with Expensive Test
LL	Log-Likelihood
MC+BN	MetaCost classifier used Bayesian network algorithm TAN as base classifier
MC+J48	MetaCost classifier used decision tree algorithm J48 as base classifier.
MDL	Minimum Description Length
MI	Mutual Information
MWST	Maximum Weight Spanning Tree
SE	Simple Estimator
TANs	Tree Augmented Naïve-Bayes networks
TN	True Negative
TP	True Positive
UML	Unified Modelling Language
Θ	Bayesian parameters
G	Graph
$P(j\|x)$	The probability estimation of classifying the instance x into class j
$\mathrm{Cost}(i,j)$	The cost of misclassification of class i
ICF_A	Information Cost Function for an attribute A
Π_{x_i}	Parents of node x_i

ACKNOWLEDGMENTS

First of all, I would like to thank ”ALLAH ALMIGHTY” who has given me the strength, patience and knowledge to continue and finish my PhD journey which started as an idea and led to a four-year-long study process. It would not have been possible to write this PhD thesis without the help and support of the kind people around me, to only some of whom it is possible to divulge particular mention here. I would like to express my deepest sense of gratitude to my great supervisor Professor Sunil Vadera for his assistance, support and feedback during this research, as well as his help in pointing me in the right direction. After four years of patience and hard work, my dream has really come true following his support. Therefore, I need to state that the congratulation compliments which I may receive should be extended to him.

Special thanks to my fabulous family, especially my parents, sisters, and brothers who have pushed themselves to the extreme ends to ensure that I continue my education to the highest level. I have a special feeling of gratitude towards my husband, who formed my vision and encouraged me in achieving my goal, through continued patience at all times. I wish to record my special thanks and gratitude to my wonderful daughters Ranim and Ratil and to my delightful son Mohammed, for being there for me throughout the entire doctorate programme. Truly, without their love and support, I would not have reached this point in my life and this PhD research work would not have been possible.

I would like to extend a huge, warm thanks to my friends, Dr. Haya Alshehri, Dr. Rabea Elmazuzi, and Dr. Majda Elferjani; they were always by my side during difficult situations and they have always supported and encouraged me.

Last but not least, I would like to acknowledge and thank everyone who helped me with good advice during the course of my research work.

LIST of PUBLICATIONS

External Publications

Nashnush, E. and Vadera, S. (2014). Cost-Sensitive Bayesian Network Learning Using Sampling. In *Recent Advances on Soft Computing and Data Mining*. Springer International Publishing, pp. 467-476.

Nashnush, E. (2014). Cost-Sensitive Bayesian Network algorithms . *Libya Higher Education Forum. "A Vision for the Future". http://libyaed.com/* . 5 - 6 June 2014, London.

Nashnush, E. and Vadera, S. (2014). Learning Cost-Sensitive Bayesian Networks via Direct and Indirect Methods. *Integrated Computer-Aided Engineering Journal*, (in process, it has been submitted on 30 September 2014).

Nashnush, E. and Vadera, S. (2015). EBNO: An Algorithm for Evolving Cost-Sensitive Bayesian Networks. *ACM journal on Knowledge Discovery from Data (TKDD)*, (in process, it has been submitted on 28 April 2015).

Internal Publications

Nashnush, E. and Vadera, S. (2012). Cost-Sensitive / Insensitive Learning algorithms. *3rd Computing Science and Engineering Post Graduate Conference*, Salford University, UK.

Nashnush, E. and Vadera, S. (2013). Direct and indirect approaches for learning cost-sensitive Bayesian network. *4th Computing Science and Engineering Post Graduate Conference*, Salford University, UK.

[*]Nashnush, E. and Vadera, S. (2013). Cost-Sensitive Bayesian Network Learning Algorithm. *Salford Postgraduate Annual Research Conference 2013 (SPARC 2013)*, 5-6 Jun 2013, University of Salford, UK.

Nashnush, E. and Vadera, S. (2013). Cost-Sensitive Bayesian Network learning using Sampling approach. *Dean's Annual Research Showcase*, Poster and abstract. Salford University, UK.

Nashnush, E. and Vadera, S. (2015). Three approaches for Cost-sensitive Bayesian Network algorithm. *A Three Minute Thesis at the 2015 Salford Postgraduate Annual Research Conference (SPARC 2015),* Salford University, UK.

Nashnush, E. and Vadera, S. (2015). Using Genetic Algorithms to optimize Tree Augmented Naïve Bayes classifier. *Dean's Annual Research Showcase.* Poster and abstract. University of Salford, UK.

[*] This poster was selected by delegates as the best poster in the conference and was awarded first prize.

ABSTRACT

Bayesian networks are becoming an increasingly important area for research and have been proposed for real world applications such as medical diagnoses, image recognition, and fraud detection. In all of these applications, accuracy is not sufficient alone, as there are costs involved when errors occur. Hence, this thesis develops new algorithms, referred to as cost-sensitive Bayesian network algorithms that aim to minimise the expected costs due to misclassifications. The study presents a review of existing research on cost-sensitive learning and identifies three common methods for developing cost-sensitive algorithms for decision tree learning. These methods are then utilised to develop three different algorithms for learning cost-sensitive Bayesian networks: (i) an indirect method, where costs are included by changing the data distribution without changing a cost-insensitive algorithm; (ii) a direct method in which an existing cost-insensitive algorithm is altered to take account of cost; and (iii) by using Genetic algorithms to evolve cost-sensitive Bayesian networks.

This research explores new algorithms, which are evaluated on 36 benchmark datasets and compared to existing cost-sensitive algorithms such as MetaCost+J48, and MetaCost+BN as well as an existing cost-insensitive Bayesian network algorithm. The obtained results exhibit improvements in comparison to other algorithms in terms of cost, whilst still maintaining accuracy. In our experiment methodology, all experiments are repeated with 10 random trials, and in each trial, the data divided into 75% for training and 25% for testing. The results show that: (i) all three new algorithms perform better than the cost-insensitive Bayesian learning algorithm on all 36 datasets in terms of cost; (ii) the new algorithms, which are based on indirect methods, direct methods, and Genetic algorithms, work better than MetaCost+J48 on 29, 28, and 31 out of the 36 datasets respectively in terms of cost; (iii) the algorithm that utilise an indirect method performs well on imbalanced data compared to our two algorithms on 8 out of the 36 datasets in terms of cost; (iv) the algorithm that is based on a direct method outperform the new algorithms on 13 out of 36 datasets in terms of cost; (v) the evolutionary version of the algorithm is better than the other algorithms, including the use of the direct and indirect methods, on 24 out of the 36 datasets in terms of both costs and accuracy; (vi) all three new algorithms perform better than the MetaCost+BN on all 36 datasets in terms of cost.

Chapter 1: Introduction

This chapter presents the thesis introduction and methodology. Section 1.1 provides an introduction of classification algorithms and Bayesian network algorithms. Section 1.2 presents the problem definition and the motivation for study. Section 1.3 presents the research questions, while Section 1.4 describes the research methodology that used. Section 1.5 explains the research hypothesis, aims and objectives and finally, Section 1.6 outlines the structure of the thesis.

1.1 Introduction

Classification is one of the most important methods in data mining, which plays an essential role in data analysis and pattern recognition, and requires the construction of a classifier. A classifier can predict the class label for an unseen instance from a set of attributes. As Friedman (1997) states:

> "*The induction of classifiers from datasets of pre-classified instances is a central problem in machine learning*".

Many methods and algorithms have been introduced to enable systems to learn classification models, such as decision trees, decision graphs, Bayesian networks, neural networks, and decision rules. In the last decade, graphical models have become one of the most popular tools to structure uncertain knowledge. *Bayesian Networks* are becoming an increasingly important area of research and are applied in several fields of artificial intelligence (Pourret et al., 2008; Kenett, 2012). There are a range of names used for probabilistic networks, including: belief networks, knowledge maps, probabilistic causal networks, causal networks, or probabilistic networks, causal probabilistic networks, Bayesian networks, Probabilistic Cause-Effect Models, and Probabilistic Influence Diagrams (Pearl, 1988). One of the most powerful characteristics of Bayesian networks is their ability to update the beliefs of each random variable via *bi-directional propagation* of new information through the whole structure.

An important feature of Bayesian networks is the way it propagates the impact of new evidence, providing each node with a belief vector that is consistent with the axioms of probability theory (Pearl, 1988; 2014). For example, the diagram in Figure 1.1 shows a simple example, presented by Pearl (2014), to model an *alarm problem* with a Bayesian network: if somebody calls you and informs you that your alarm has gone off, you might think there is a burglar in your home, and you will go to your home directly. On your way, if you hear a radio announcement that there was an earthquake nearby, you might reconsider given that the earthquake may have caused the alarm. In particular, from this information, the BNs can propagate the impact of evidence from *effect to cause* (Radio →Earthquake), then from *cause to effect* (Earthquake → Alarm), and then again from *effect to cause* (Alarm → Burglary). In this figure, A represents Alarm and B represents Burglary, the impact of the evidence from the Radio announcement will be to update the beliefs so that A→B less credible.

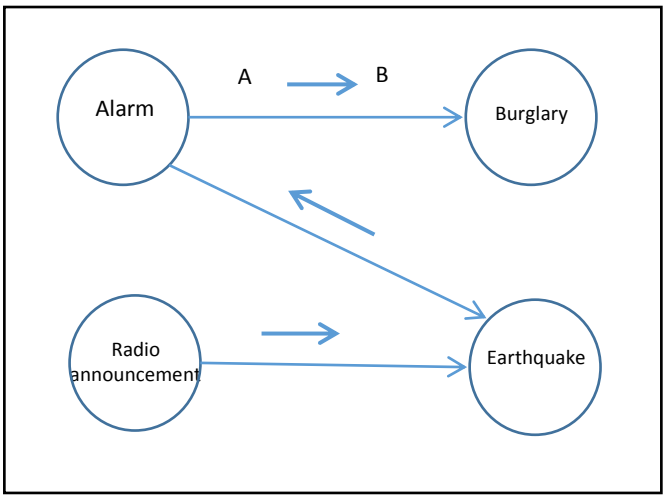


Figure 1.1: Propagation and the impact of evidences (Pearl, 1988; 2014).

Over the last few years, Bayesian networks have become very popular. Bayesian networks and their algorithms are explained by Pearl (2001), who won the Association for Computing Machinery Turing Award in 2012. Moreover, Bayesian networks have been successfully applied in different areas to create consistent probabilistic representations of uncertain knowledge in several fields, including: medical diagnosis (Spiegelhalter et al., 1989; Heckerman et al., 1995), image recognition (Booker and Hota, 2013), language understanding (Charniak and Goldman,1989), search algorithms (Hansson and Mayer, 1989). In particular, the book by Pourret et al. (2008) and Kenett (2012) describes 21 applications of Bayesian networks to illustrate their wide range of applications in clinical decision support,

complex genetic models, crime risk factor analysis, inference problems in forensic science, terrorism risk management, credit rating of companies, and enhancing human recognition.

In machine learning algorithms, several studies have mentioned that, learning processes should take account of the costs involved in decision-making (Breiman et al., 1984; Turney,1995, 2000; Zadrozny, and Elkan, 2001). Turney (2000) lists the kind of costs that should be considered, such as cost of misclassification, the cost of test, the computational cost, data acquisition cost, active learning cost, human computer interaction cost, and cost of teacher. Amongst these, the misclassification cost is one of the most important. In fact, misclassification cost happens when, the examples that belong to negative class are classified to positive class (FP; classifying a negative example as positive), or the examples that belong to positive class are classified to negative class (FN; classifying a positive example as negative). For example, in a credit card fraud detection application, if the system classifies a transaction of a customer as a non-fraud when fraud has occurred, it is likely to result in financial loss. In contrast, if a system classifies a transaction as a fraud when it is not the costs would involve some further checks before proceeding with the transaction.

This observation has led to many recent studies focusing on cost-sensitive learning algorithms. Historically, most of the cost-sensitive algorithms developed have focussed on learning decision trees, with a recent survey comparing over 50 algorithms (Lomax and Vadera, 2013). In contrast, little attention has been paid to developing cost-sensitive Bayesian networks (Gao, et al., 2008; Nashnush and Vadera 2014; Jiang and Wang, 2014; Kong et al. 2014). Hence, the main focus of this thesis is to study whether it is possible to develop a new machine learning algorithm to learn Bayesian Networks that can perform cost-sensitive classifications.

1.2 Motivation

Inductive learning techniques have been used successfully to build classifiers and obtain good prediction results in a number of applications, including Customer Target Marketing (Rygielski et al., 2002), Medical Disease Diagnosis (Cios and Moore, 2002), Supervised Event Detection (Zhang et al., 2010), Multimedia Data Analysis (Kantardzic, 2011), Biological Data Analysis (Bishop, 2006), and Social Network Analysis (Aggarwal,2014). In

particular, in traditional machine learning classification algorithms such as decision tree induction, neural networks, Bayesian networks, the aim is to build a model using a training set, and then use the model for classifying unseen cases. Figure 1.2 shows such an example, where some training data is used as input to a learning algorithm, which classifies whether there has been a fraudulent transaction. Historically, most of these techniques only focus on predicting correct results and maximising accuracy. More recently, as mentioned above, there has been recognition that costs play an important role and should be taken into account when developing classification algorithms. In particular, in real world applications, one should take into consideration misclassification costs (Turney,2000).

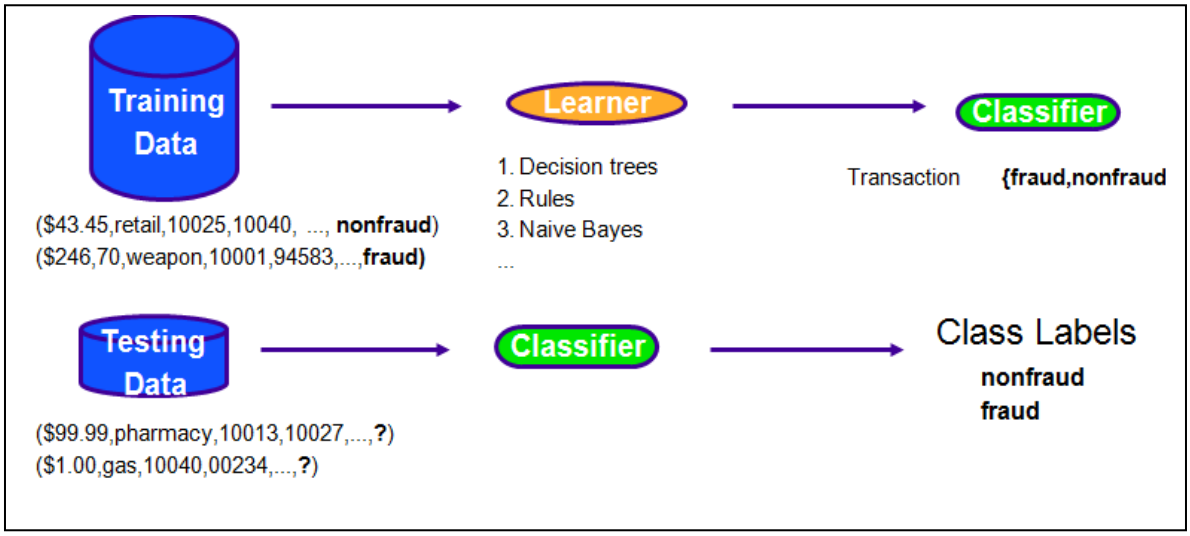


Figure 1.2: How cost-insensitive classification algorithms work (Fan et al., 2002).

Cost-sensitive learning is a type of learning in data mining that takes account of costs such as misclassification costs, test costs, or any other costs into consideration (Turney, 2000), and aims to minimize total costs by treating the different classification errors differently (Ling et al., 2006). On the other hand, c*ost-insensitive learning*, does not take the misclassification costs into consideration and focuses on accuracy only.

Therefore, the performance of any AI application should be balanced between accuracy and cost, as through accuracy alone is not enough. In particular, in real world problems, the data is imbalanced, where the most expensive errors tend to be associated with the rare cases, while the most cheapest errors tend to be associated with the frequent cases, and a learner will learn from very highly skewed data, thus, a cost-insensitive classifier that aims to

increase the accuracy will be biased to classify instances to most frequent case (He and Garcia, 2009). The following examples illustrate the need to take account of costs:

- To detect a fraudulent customer, the cost of misclassifying a customer who commits fraud (rare class) is greater than the cost of misclassifying a customer who is non-fraudulent (common class).
- Also, in a medical application, the cost of misclassifying a patient who has cancer is greater than the cost of misclassifying a patient who does not have cancer.

In these types of domain, building a classifier that does not consider the cost of misclassification is unlikely to perform well because it will be biased towards the instances under the category of the frequent class, which will result in producing a useless classifier. Thus, cost-sensitive learning algorithms that take costs into consideration and deal with different types of cost differently are essential (Charles and Victor, 2008).

Hence, a number of authors, who recognised the need for taking account of costs, have focussed on developing cost-sensitive decision tree learning algorithms, including : *Cost-Minimization* (Pazzani et al., 1994), *Decision Tree with Minimal Costs* (Ling et al., 2004), *EG2* (Núñez, 1991), *CS-ID3* (Tan and Schlimmer, 1989), *IDX* (Norton 1989), *CS-C4.5* (Frietas et al., 2007), *CSNL* (Vadera, 2010). All of these algorithms use the cost directly during the algorithm process. In contrast, some of the algorithms use the cost indirectly, before and after using the algorithm, such as *Costing* (Zadrozny et al., 2003b), *C4.5CS* (Ting, 2002), *MaxCost* (Margineantu and Dietterich, 2003), *MetaCost* (Domingos, 1999), *CostSensitiveClassifier* (CSC) (Witten and Frank, 2005),and *AdaCost* (Fan et al., 1999).

Although Bayesian networks have been successfully applied, there has been little, but no research on optimising them for cost-sensitive learning. Hence, this thesis explores the potential for learning Bayesian networks for cost-sensitive classification.

1.3 Research aims and objectives

The primary hypothesis of this research is that, it is possible to develop an algorithm to learn cost-sensitive Bayesian networks, which are more cost-effective on average than current

algorithms, including existing cost-sensitive decision learning tree algorithms, existing cost-sensitive Bayesian network learning algorithms, and existing cost-insensitive Bayesian network learning algorithms.

To check this hypothesis, this research aims to develop methods that learn Bayesian networks that take account of misclassification costs and then utilise empirical methods to assess the extent to which the hypothesis is true. The specific research objectives are:

1. To review the background of Bayesian networks learning algorithms, and analyse the types of this algorithm.
2. To review the literature on cost-sensitive learning, analyse the most significant issues in current cost-sensitive learning algorithms, and identify the strategies used.
3. To develop new cost-sensitive Bayesian network learning algorithms that aim to overcome the issues identified, and are based on methods of cost-sensitive learning algorithms such as direct, indirect, and optimization methods.
4. To evaluate the new algorithms against existing cost-sensitive algorithms and measure performance, and compare the algorithms in terms of accuracy, and cost minimization.

1.4 Research questions

Given the above aims and objectives, the following key questions need to be addressed when attempting to design algorithms to learn Bayesian networks that take account of costs. In relation to the research aims and objectives, each question is answered in Section 1.3:

Q1. How can a learning Bayesian algorithm involve misclassification costs?

This question is answered in objectives 1, 2 and 3 by analysing Bayesian networks algorithm, and based on the methods that used to involve costs into decision trees algorithms. Hence, the new Bayesian networks algorithms can involve misclassification costs in three different methods; direct, indirect, and optimization method.

Q2. At which stage should Bayesian networks include these costs: before construction, during construction, during learning parameters or after final construction?

This question is addressed in objectives 1, 2 and 3 by analysing the steps of existing

Bayesian networks algorithm (learning structure, and learning parameters), and based on the ways that used in cost-sensitive decision trees algorithms. Hence, new Bayesian networks algorithms can include misclassification cost before construction by using sampling approach; or during learning structure and parameters by using amending approach.

Q3. How can the costs be balanced against the need to maintain the accuracy rate?

This question is answered in objectives 3, and 4 by including the costs in the right place without changing the performance of the algorithms then evaluate these algorithms against existing cost-insensitive and sensitive algorithms.

Q4. What are the weaknesses of existing cost-sensitive Bayesian algorithms?

This question is addressed in objectives 2 by analysing the most significant issues in current cost-sensitive Bayesian networks algorithms.

1.5 Research methodology

This section describes the research methodology that used in this research, and shows the outline of the methodology adopted in this thesis. As Rajasekar et al. (2006) describe, there is a difference between *research methods*, and *research methodology*. Essentially, research methods represent all the methods, procedures, and schemas, which are used by a researcher during a research study. For example, these methods might be collecting and sampling data, using some hypotheses, and finding a solution to a problem. Also, any research that is based on experiments requires collection of facts, measurements, hypotheses, and observations, and these are called scientific research methods. Given the nature of this thesis, which is focussed on objective quantitative measures (Rajasekar et al., 2006; Kothari, 2011), this PhD research uses the quantitative research methodology because it is based on testing new hypotheses.

The main phases of the research methodology used in this study are shown in Figure 1.3, where these phases are followed to achieve the research objectives:

1. Starting with reviewing the background of Bayesian networks. The objective 1 can be achieved in this phase.
2. Identify the alternative cost-sensitive methods by studying the literature review of existing cost-sensitive algorithms that based on three methods; indirect; direct, and optimization method. Where, the objective 2 can be achieved in this phase.
3. Design new algorithms that aim to minimize misclassification costs, these algorithms are based on three methods that show in phase 2.
4. Implement CS-BN algorithms, where, this study used the open source algorithms in the data mining system WEKA, which were developed by Hall et al. (2009). The algorithms are implemented in *java NetBeans*.
5. The empirical evaluation methodology adopted to split the datasets into 75% for training and 25% for testing, and to apply the algorithms10 times randomly with 16 misclassification costs from 1 to 4 for each class label. Then, the average performance of each algorithm with standard errors are calculated 10 times (Gurland and Tripathi, 1971).
6. Test and analyse the algorithm's performance and reliability; to test the algorithms, benchmark datasets from UCI repository datasets (Asuncion and Newman, 2007) have been used to simulate problems of cost-insensitive algorithms.
7. The algorithms are modified to improve the performance. These algorithms are modified throughout the study, feedback from the supervisor, examiners, assessments, conferences, and journals have been taken into account. Hence, the objectives 3 can be achieved in phases 3 ,4, 5,6, and 7.
8. Evaluate the algorithms and compare them with existing cost-sensitive and insensitive algorithms such as MetaCost+J48, MetaCost+BN, and cost-insensitive Bayesian networks algorithm. Hence, the objectives 4 can be achieved in this phases.

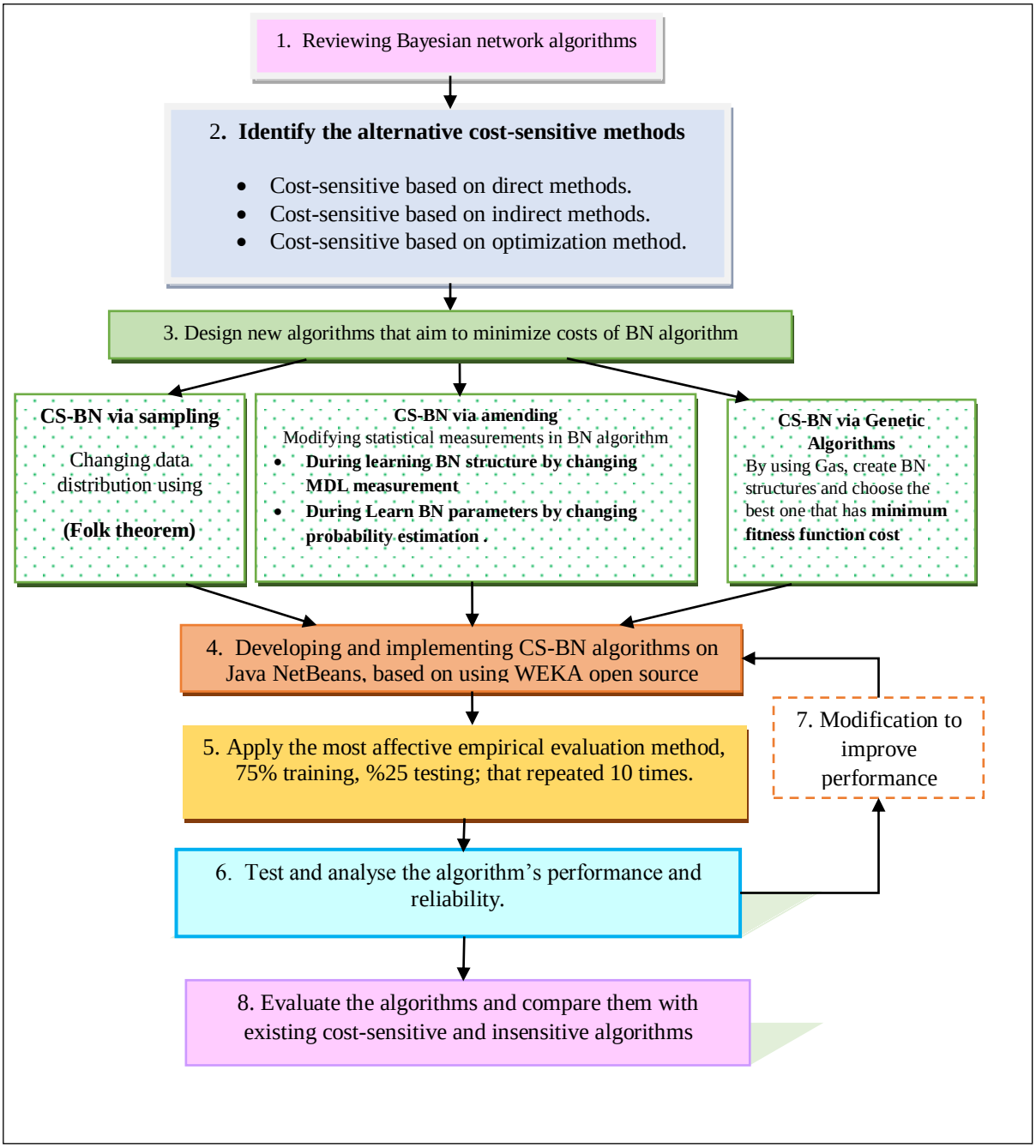


Figure 1.3: Research methodology.

1.6 Thesis organisation

The thesis is organised into six chapters, set out as follows:

Chapter 1: Thesis introduction and methodology

This chapter presents a brief introduction to the research, introducing the reader to the problem statement and motivation, potential contribution, research methodology, research hypothesis, aims and objectives.

Chapter 2: Background on Bayesian networks

This chapter presents the basic of data classification process, the background to Bayesian network learning algorithms and basic laws of probabilities. It shows types of Bayesian network algorithms with examples, how they learn a BNs structure, and how they learn the parameters.

Chapter 3: Survey of existing cost-sensitive algorithms

This chapter includes a survey of existing cost-sensitive learning algorithms; it shows the categories of cost-sensitive learning algorithms, direct, indirect, and optimization algorithms, and literature review in cost-sensitive Bayesian network algorithms.

Chapter 4: The development of cost-sensitive Bayesian network learning

This chapter presents the development of three new algorithms for learning cost-sensitive Bayesian networks; these algorithms based on, (i) indirect methods by changing the data distribution to reflect the costs, (ii) direct methods by amending an existing algorithm, (iii) optimization method by using Genetic algorithms to create a BN structures that has minimum fitness function cost.

Chapter 5: An empirical evaluation of the new algorithms for learning cost-sensitive Bayesian networks

This chapter presents a comprehensive empirical evaluation, including a comparison with existing cost-sensitive/ insensitive learning algorithms, and finally, evaluating and analysing their performance by using the average cost and accuracy rates as measurements.

Chapter 6: Conclusions and future works

This chapter summarises the aims of this work and concludes with the achievements, including reflections on the extent to which the research objectives have been met and future developments that may be necessary.

Bibliography: It presents all the references in this thesis.

Appendix: It presents real examples to learn BNs, and class implementations diagrams of our java code.

Chapter 2: Background on Bayesian Networks

This chapter presents an overview of Bayesian networks and the basic laws of probabilities. Section 2.1 describes the basics of the data classification process. Section 2.2 presents an overview of Bayesian networks, while Section 2.3 presents the principles of Bayesian networks such as probability, and inference. Section 2.4 presents algorithms for learning Bayesian networks. Finally, a summary of the chapter is presented in Section 2.5.

2.1 Data Classification

Data mining is an active research area involving the development and analysis of algorithms for extracting interesting knowledge and patterns from real-world datasets and summarizing it into useful information (Witten and Frank, 2005). Classification is one of the most important methods in data mining which plays an important role in data analysis, pattern recognition, and decision making (Aggarwal, 2014).

Classification requires the construction of a model that can be used to predict a class label for an unseen instance from a set of attributes. Classification algorithms attempt to learn the relationship between a set of variables (features) and a class label (target variable). In particular, classification algorithms learn from training instances to construct a model; where each instance is associated with a known class label. Then, in a testing phase, the model can be used to assign labels to unlabelled test instances (Aggarwal, 2014). Figure 2.1 shows how the classification process can be divided into two steps:

i. *Model construction*: training data is used to create a model, where the model is represented in some forms such as classification rules, decision trees, Bayesian networks, or mathematical formulae.

ii. *Model usage*: the model is used for classifying unseen or unknown instances, and estimating the accuracy of the model based on the known class label of test instance.

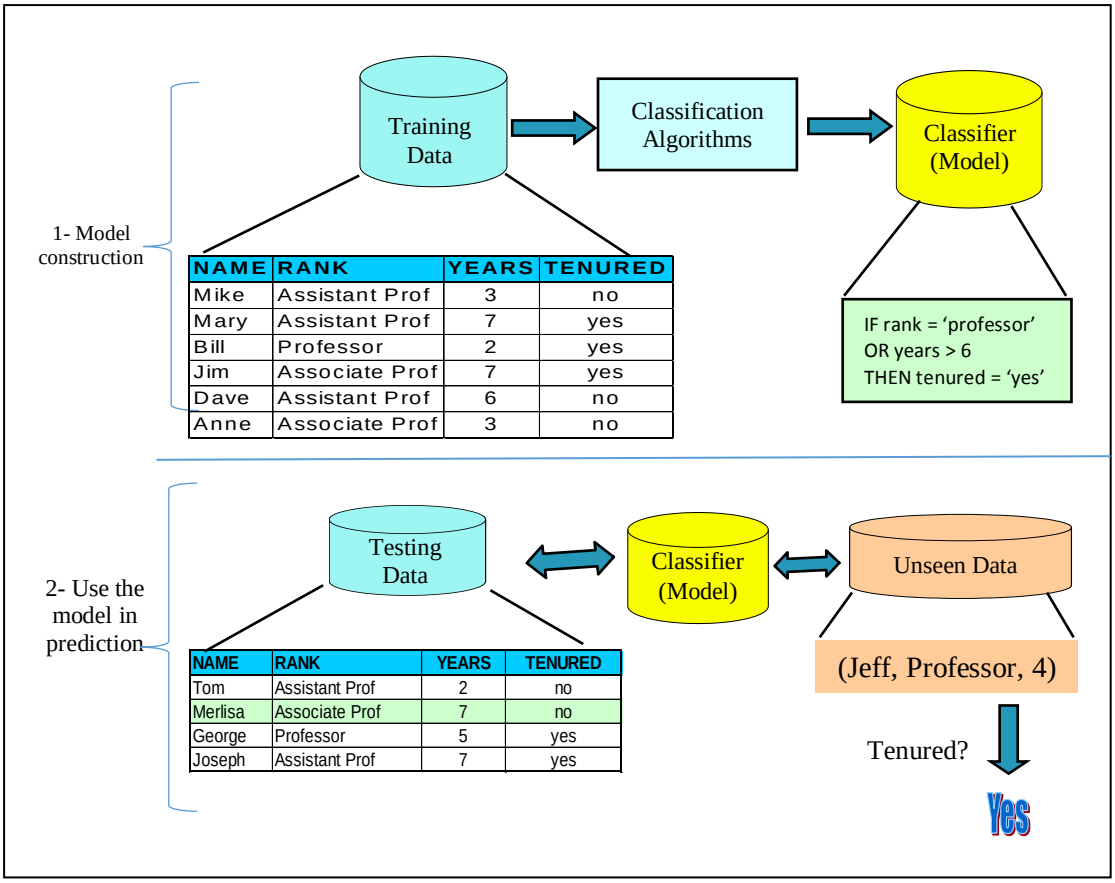


Figure 2.1: Classification process (Han et al., 2015).

Classification algorithms have been used in several applications, such as: customer target marketing (Rygielski et al., 2002), medical disease diagnosis (Cios and Moore, 2002), supervised event detection (Zhang et al., 2010), multimedia data analysis (Kantardzic, 2011), biological data analysis (Bishop, 2006), and social network analysis (Aggarwal, 2014). There are several techniques used for data classification such as:

- *Decision trees algorithms* that use a decision tree that is learned from labelled training instances (Quinlan, 1986).
- *Rule-based algorithms* for classifying examples using a collection of "*if -then*" rules (Cohen,1995).
- *Instance based algorithms* that perform classification using only specific instances (Aha et al., 1991).
- *Neural networks algorithms* that use a computational model based on biological neural networks (Funahashi, 1989).

- *Bayesian networks algorithms* that are statistical classifiers and are based on Bayes theorem (Pearl, 1988; 2014).

This thesis focuses on Bayesian networks, and hence the following sections describe the foundation for Bayesian networks. Section 2.2 provides an introduction to the main concept, Section 2.3 describes the main principles of probabilities used when performing classification, and Section 2.4 describes algorithms that learn Bayesian networks.

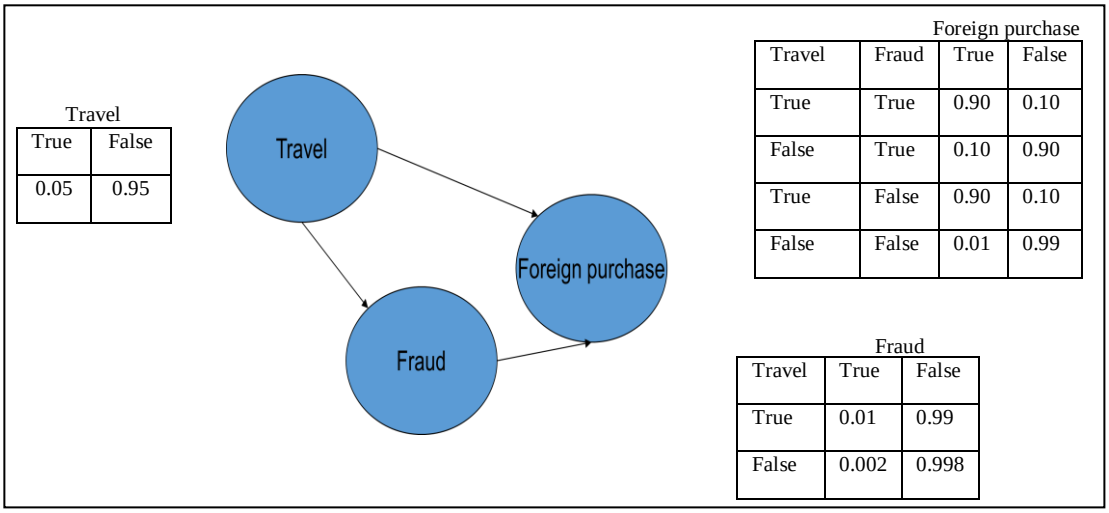
2.2 Overview of Bayesian networks

Bayesian networks, which were invented in 1988 by Judea Pearl, changed the focus of AI from logic to probability. In the last decade, Bayesian networks have become one of the most popular tools to structure uncertain knowledge. Indeed, Bayesian networks have been successfully used in a number of fields including medical diagnostic systems (Spiegelhalter et al., 1989**;** Heckerman et al., 1995), in NASA AutoClass project for data analysis and control the space shuttle (Morris, 2003), Fraud detection systems(Maes et al., 2002), and Speech recognition systems (Zweig and Russell, 1998).

A Bayesian network can be used as a classifier by computing a posterior probability of a set of labels given the observable features and the classifier classifies new instance according to the probability of the class label (Sebe et al., 2005). In particular, a Bayesian network classifier aims to find the class that has the highest probability given an observed case (Salama and Freitas, 2013). According to Heckerman (2008) Bayesian networks have several advantages for data modelling. Firstly, the model of BNs encodes dependencies among all nodes and it can handle situations where some data entries are missing. Secondly, Bayesian statistical methods offer an efficient and principled approach for avoiding the *overfitting* of data. Thirdly and finally, Bayesian networks do not need to determine the full joint distributions, which will be described later in Section 2.3, as they merely determine local conditional distributions and the network can automatically represent the joint distribution. Bayesian networks are powerful tools for knowledge representation and inference that encode dependence and independence relationships between variables. In particular, a Bayesian network model is a probabilistic model that represents variables (continues or discrete) of data as nodes, and the correlations between these nodes represents the joint probability distribution between variables (nodes). Obviously, the edges between nodes represent

(in)dependence between nodes that will be described later in Section 2.3.1). Where a direct edge represents the direct influence between nodes (statistical dependency), while an indirect edge of nodes that are not connected, represents the indirect influence between nodes (statistical conditional independency) (Corani et al., 2012), where direct and indirect influence will be described later in Section 2.3.2.2.

More specifically, in BNs' structures each node has a set of values, and the relationship between the node and its parents is defined by a *conditional probability table* (CPT). This table determines the probabilities of the values between a stated node given its parents. For example, Figure 2.2 shows a simple fraud detection Bayesian network, with CPTs of fraudulent transactions which are more likely to happen when the card holder is travelling abroad because tourists are targets for thieves, as travel and fraud are causes for foreign purchase. Invariably, travel explains foreign purchase, thus is evidence against fraud, while the network has three nodes, representing Travel, Fraud, and Foreign purchase, respectively. The travel node, as being a parent node has a prior probability table that indicates the chances of someone travelling to be 0.05 and not travelling to be 0.95. Additionally, the table for the Fraud node shows the probability of fraud given values of its parent node, Travel. Thus, the probability of fraud for someone travelling is 0.01, and 0.002 if it is not travelling. While, the probability of no fraud for someone travelling is 0.99, and 0.998 if not travelling. This is very similar to the Foreign Purchase node.



Travel

True	False
0.05	0.95

Foreign purchase

Travel	Fraud	True	False
True	True	0.90	0.10
False	True	0.10	0.90
True	False	0.90	0.10
False	False	0.01	0.99

Fraud

Travel	True	False
True	0.01	0.99
False	0.002	0.998

Figure 2.2: A simple Bayesian network for fraud detection (Ezawa and Schuermann, 2015)

Given such a network, it can be used to update when evidence is made available. For example, if one knows that a person is travelling (Travel is True), the probabilities of Fraud

given Travel to be true and false become 0.01 and 0.99 respectively. Also, the Foreign Purchase node is updated to 0.90, and 0.10 when the Foreign Purchase are true and false respectively.

2.3 Principles of Bayesian networks

This section describes the key principles of Bayesian networks, while, Section 2.3.1 summarises some definitions from probability theory including Bayes rule which is central to Bayesian networks, and also presents the notions of dependence and independence. Section 2.3.2 explains the basic of BNs, also how the information is propagated in BNs, and shows how to use statistical inference based on the Bayes rule to update the probability for a hypothesis as evidence is acquired.

2.3.1 Definitions from probability theory

This section describes the basic laws of probabilities and shows how to calculate the probability distribution between two events based on whether they are *dependent* or *independent* events. A *probability* function P(A) of an event A, represents the density function of A, while, a *joint probability* P(A,B) is the probability of two events, A and B, occurring together at the same time.

2.3.1.1 Dependency events

Formally, if two events are dependent, namely they do influence each other in any way, then:

$$P(A, B) = P(\mathrm{A} \cap \mathrm{B}) = \mathrm{P(A)} * \mathrm{P(B\ after\ A)} \qquad (2.1)\ \text{where A, and B are dependent}$$

In particular, if the two events are considered dependent, then the outcome of the one event depends on the probability of the other event (Ben-Gal, 2007). For example, if one has a bag that contains 4 balls green, 2 balls red, and 1 ball blue, where in each time we have to choose one ball without replacement, then each event is dependent on the other events as illustrated in Figure 2.3, and according to equation (2.1) the probability of choosing green and red is:

$$\mathrm{P(Green, Red)} = \mathrm{P(Green)} * \mathrm{P(Red\ after\ Green)} = \frac{4}{7} * \frac{2}{6} = \frac{8}{42} = 0.19$$

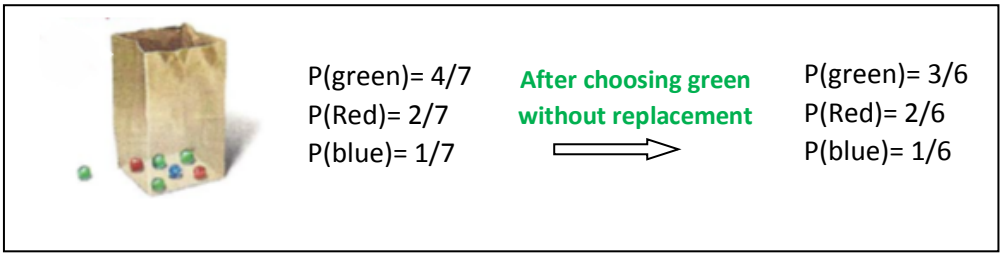


Figure 2.3: Illustration of three dependency events (Sawaal, 2015).

2.3.1.2 Independency events

Formally, if two events are independent, namely they do not influence each other in any way, then:

$$P(A,B) = \mathrm{P(A)} * \mathrm{P(B)} \qquad (2.2) \text{ where A, and B are independent}$$

If the two events are considered independent, then subsequently each can occur individually and the outcome of one event does not rely on the other. Hence, this will occur if the fact A occurring does not affect the probability of B occurring (Ben-Gal, 2007). For example, this can be noted if one has 2 events; choosing a random card from 5 cards, and rotating a wheel has 8 parts, where both of events are independent. According to equation (2.2), the probability of choosing card number 10 and rotating a wheel on part 6 is:

$$\mathrm{P(Card\ 10, Wheel\ on\ 6)} = \frac{1}{5} * \frac{1}{8} = \frac{1}{40} = 0.025$$

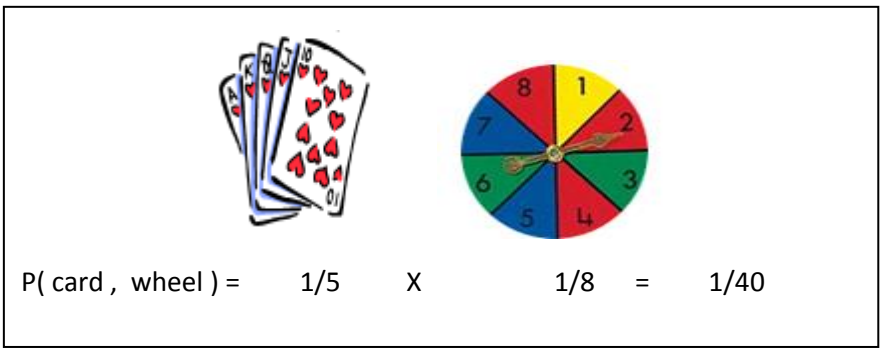


Figure 2.4: Illustration of two independency events (Sawaal, 2015).

2.3.1.3 Conditional probability

If two events are dependent, then we have to use the concept of conditional probability. Conditional probability is the probability of an event (A) occurring, given that another event (B) has already occurred. The conditional probability reduces the sample space of giving the outcome. Formally, conditional probability can be defined by:

$$P(A|B) = \frac{P(A,B)}{P(B)} \qquad (2.3) \quad \text{where A, and B are dependent.}$$

$$P(A|B) = P(A) \qquad (2.4) \quad \text{where A, and B are independent.}$$

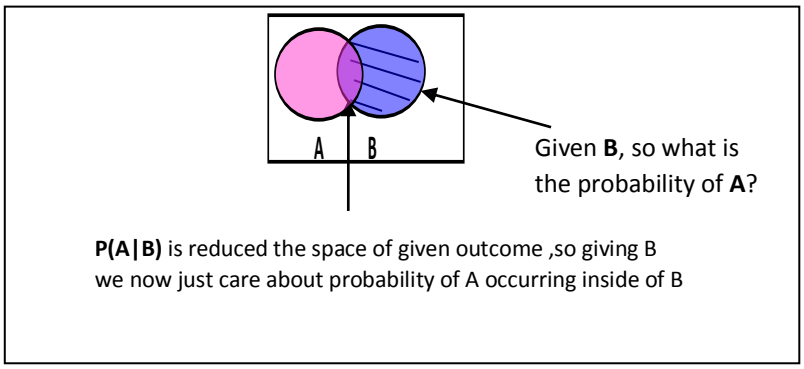


Figure 2.5: Conditional probability example (Kountz et al., 2011).

Bayes's theorem was introduced by Thomas Bayes (1701 - 1761) and represents how the conditional probability of a set of possible causes for given an observed outcome. In particular, this theorem is used for statistical inference (Bolstad, 2013), and it is stated mathematically as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \qquad (2.5)$$

Where:

- A and B are events, and B is *observed*.
- P(A) is *prior probability*
- P(B) is *observed probability*.
- P(B|A) is a *likelihood probability*; the conditional probability of B given that A is true.
- P(A|B) is a *posterior probability*; the conditional probability of A given that B is true, it reflects the belief about the hypothesis after B has been observed.

For example, to calculate the probability of someone who has brown hair and given female, when given the Table 2.1:

Total = 11	Female	Male
Brown hair	3	4
Blond hair	2	2

Table 2. 1: Joint probability example.

$$\mathrm{P}(\text{Brown hair} \mid \text{Female}) = \frac{\mathrm{P}(\text{Brown hair} \cap \text{Female})}{\mathrm{P}(\text{Female})} = \frac{3/11}{5/11} = \frac{3}{5} = 0.6$$

2.3.2 Bayesian networks structure

This section presents the concept of Bayesian networks; where Section 2.3.2.1 shows how to use BNs model joint distributions of a set of variables, and how BNs use conditional probabilities between nodes (variables) to compute the probability of events. It presents the *Chain theorem* which is used to calculate the joint probability distribution over sets of random variables in the BNs structure. Section 2.3.2.2 demonstrates how to use *Bayes's theorem* to enable *inference* when certain pieces of evidence are available to answer queries and update beliefs.

2.3.2.1 Bayesian networks basics

A Bayesian network is a probabilistic model that represents variables (continues or discrete) of data as nodes in a *Directed Acyclic Graph* (DAG), and the relationships between these nodes represents the joint probability distribution between nodes. Edges between nodes represent the direct correlations between variables (Corani et al., 2012). For example, if we have two nodes {A, B} are present with the edge from node A to node B, being relevant, then A has a direct influence on B, where these directed edges between the nodes represent probabilistic dependencies among the corresponding random variables. In fact, BNs represent a model of the joint probability distribution of n random variables $\mathrm{X} = \{x_1, x_2, x_3, \ldots, x_n\}$, and the edges in a network represent the conditional (in)dependencies among the nodes. Whereas, each node has a set of values with the parent nodes, and it gives the probability of the variable represented by the node in a conditional probability table (CPT). This table determines the probability of all parents of a node which are affected by other nodes; where the CPT is computed from data and it represents the frequency of events in dataset. In addition, CPTs will be described later in BNs parameter learning, Section 2.4.3.

Formally, a Bayesian network is represented as DAG that encodes a joint probability distribution over a set of random variables X. This shows as a pair of graph G and parameters Θ B=<G, Θ>, where G is a DAG of n random variables $X = \{x_1, x_2, x_3, \ldots, x_n\}$, and the graph G encodes independence assumptions; each variable x_i is independent of its non-descendants given its parents in G. While, Θ represents the set of parameters between the nodes. In particular, a parameter of each node x_i in X, is represented as $P(x_i|\Pi_{x_i})$, where Π_{x_i} is the set of parents of node x_i. More precisely, a BN uses a chain theorem to calculate the joint probability distribution over sets of random variables, as demonstrated in equation (2.6). It is best to let a BN be a Bayesian network over variables, $X = \{x_1, x_2, x_3, \ldots, x_n\}$, as the BN specifies a unique joint probability distribution P(X) given by the product of all conditional probability tables specified in the BN (Schum, 2001). Given that, by definition, each node x_i has a conditional probability distribution with its parent $P(x_i|\Pi_{x_i})$, and the chain rule can be used to define the joint distribution as follows:

$$P(x_1, x_2, x_3, \ldots, x_n) = \prod_{i=1}^{i=n} P(x_i|\Pi_{x_i}) \qquad (2.6)$$

For example, the network in Figure 2.2 (the Fraud example), can be used to model the joint distribution and to find what is the probability if someone is travelling, and will not receive a fraudulent transaction, and he will make foreign purchases. P(Travel=True, Fraud=False, Foreign Purchase=True) = P(Travel)*P(Fraud|Travel)*P(Foreign purchase| Travel,Fraud) = 0.05 * 0.99 * 0.90= 0.0445 .

More precisely, *inference* in a Bayesian network involves updating the probabilities of nodes' given evidence and is described in the following section.

2.3.2.2 Bayesian inference

In a Bayesian network structure, some variables can be observed, where these observations can update the new information in the structure, and the process of conditioning is called *inference*, where it involves the propagation or revision of probabilities on the domain of the structure. In particular, there are four types of inference, which are based on query, and evidence nodes. To illustrate the types of inference, it is necessary to consider Figure 2.6 which is a modified version of the so-called "Asia" problem (Lauritzen and Spiegelhalter, 1988) that is also one of the examples used in a Bayesian networks tool known as *Netica* .

This network model is part of the *lung cancer* problem and can be used in scenarios, such as with a patient who visits a doctor with breathing difficulties (known as Dyspnoea) and is worried that he has lung cancer. A doctor also knows that other relevant information that increase the chances of cancer such as pollution, and smoking, as well as, a positive X-ray would indicate lung cancer. Consequently, through this scenario, there are four types of inference, as shown in Figure 2.7; where E is evidence node, and Q is query node:

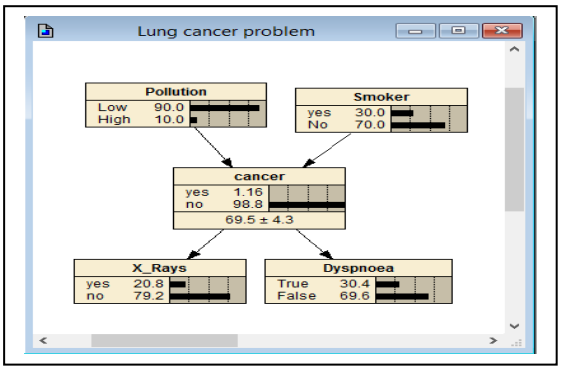


Figure 2.6: BNs' structure of lung cancer problem using *Netica.*

i. *Diagnostic inferences (inference from effect to cause):* This type of inference starts from effects to causes, and occurs in the opposite direction to the arcs, from effects to causes. For instance, in the above example, if one observes Dyspnoea, then, as illustrated in Figure 2.7(a), evidence propagates from symptoms Dyspnoea to Cancer, and then up to Pollution and Smoker, the results in propagation down from cancer to X-Rays (Korband and Nicholson, 2010). Comparatively, the process of going up from a child to its parents is illustrated in Figure 2.7(a).

ii. *Causal inferences (inference from cause to effect):* This type of inference starts from cause to effects as illustrated in Figure 2.7(b) where evidence is provided that a person smokes, then this is propagated down the arrows, from Smoker to Cancer, then to X_Rays and Dyspnea. The change in the probability of Cancer also results in propagation up to Pollution. Whereas, the process of going down from a parent to children, as illustrated in Figure 2.7(b), is known as causal inference.

iii. *Intercausal inferences (inference between cause and common effects):* This type of inference starts from cause to cause through effects, where both causes are independent of each other as illustrated in Figure 2.7(c), where evidence is provided that a person smokes, and this is propagates down the arrows, from Smoker to Cancer and then propagated up to pollution. The change in the probability of Cancer is subsequently affected through the results in propagation up to Pollution. Whereas, the process of

going from a parent to parent through its children as illustrated in Figure 2.7(c), is known as intercausal inference.

iv. Mixed Inferences: This type of inference is mixed between different types of inferences, where any node might be a query or piece of evidence, thus this inference can combine the above types of inference, as shown in Figure 2.7(d).

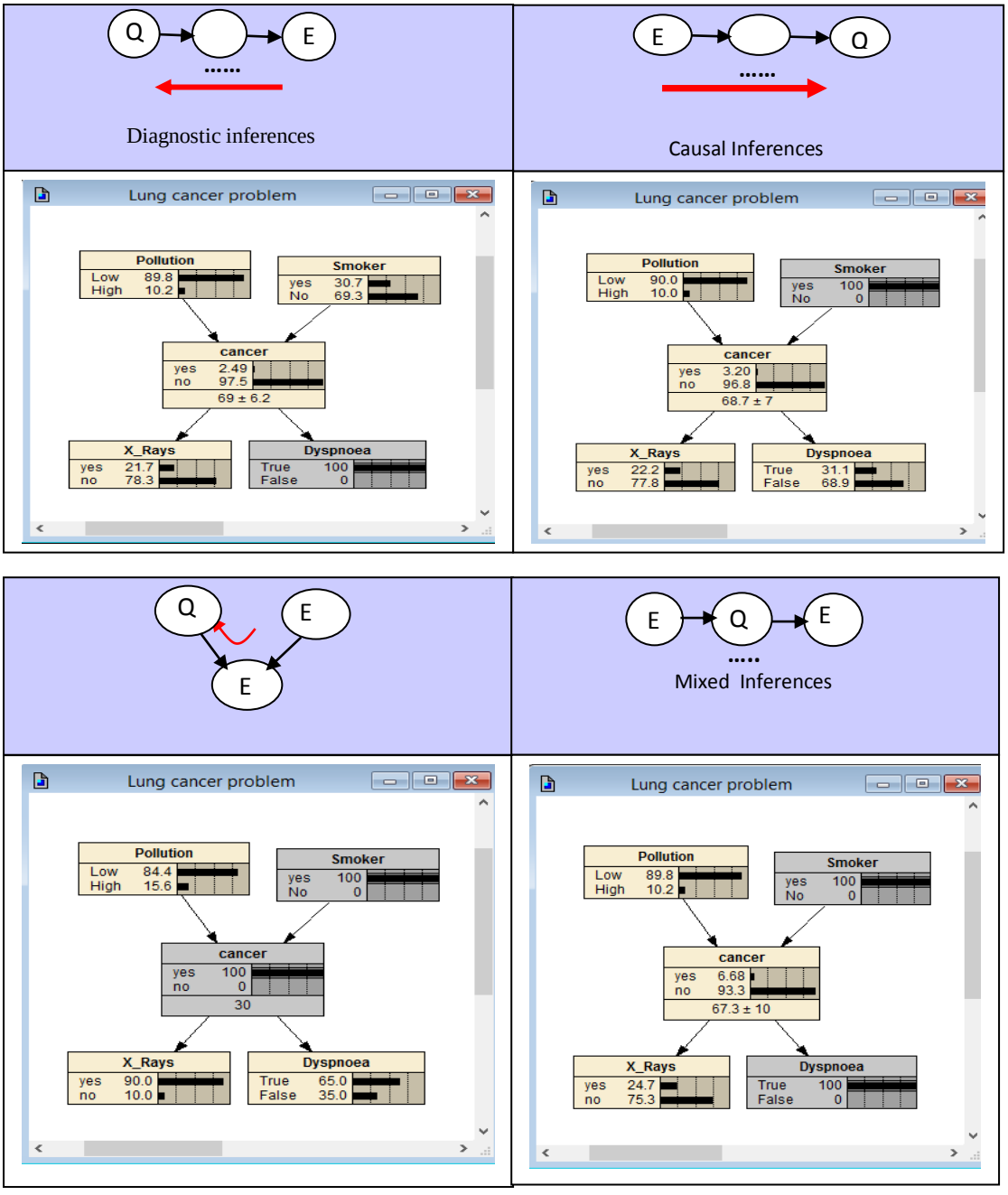


Figure 2.7: Types of inferences

2.4 Learning Bayesian networks

A Bayesian network can be used as a classifier by computing the posteriori probability of a set of labels given the observable features (Pearl, 1988), where to build a complete BN classifier, there are two aspects to constructing a BN (Neapolitan, 2004):

- *learning the graphical structure* (topology), which studies the qualitative part and how to find a graphical relationships between the variables.

- *learning the parameters* (conditional probability estimation), which studies how to quantify the relationships and how to determine the extent of the relationship between the variables and takes the form of a table that represents the conditional probabilities between a node and in its parents in CPT.

2.4.1 Bayesian network structure learning

Several algorithms have been developed to learn the structure of a Bayesian network. One of the first methods was due to Chow and Liu (1968), who introduced an algorithm for learning *a Bayesian tree* based on approximating the joint distribution of a set of attributes by using the distributions that involving no more than pairs of attributes as shown in Figure 2.8(a). Duda and Hart (1973) and Langley et al. (1992) proposed an algorithm for learning a simpler structure known as a *Naïve Bayes* structure, where all attributes are represented as independent nodes that have one parent node which represents the class (Langley 1992). Figure 2.8(b) shows an example Naïve Bayes network, where, a Naïve Bayes classifier assumes conditional independence of the features given the class, and it is easy to construct and it has been used as a classifier for many years, especially where the features are not strongly correlated. Pearl (1988) developed an algorithm to learn singly-connected graphs, which are *Directed Acyclic Graphs* (DAGs) where any two nodes have at most one unique path between them as shown in Figure 2.8(c). More recently, Friedman et al. (1997) have developed a natural extension to the Naïve Bayes classifier and the Chow-Liu algorithm, where they introduce the *Tree Augmented Naïve-Bayes* (TAN) structure. In contrast to Naïve Bayes, where the assumption is that all attributes are independent, a TAN can model all dependencies between attributes by allowing the attributes to form a tree. Thus in a TAN structure, the correlations between attributes can be captured by adding additional edges

between attributes as shown in Figure 2.8(d). Given that a TAN improves upon Naïve Bayes by avoiding its conditional independence assumptions, avoids the computational overhead of a general Bayesian network, and has been shown to be an effective classifier (Friedman et al., 1997), thus, we adopt TANs in this study.

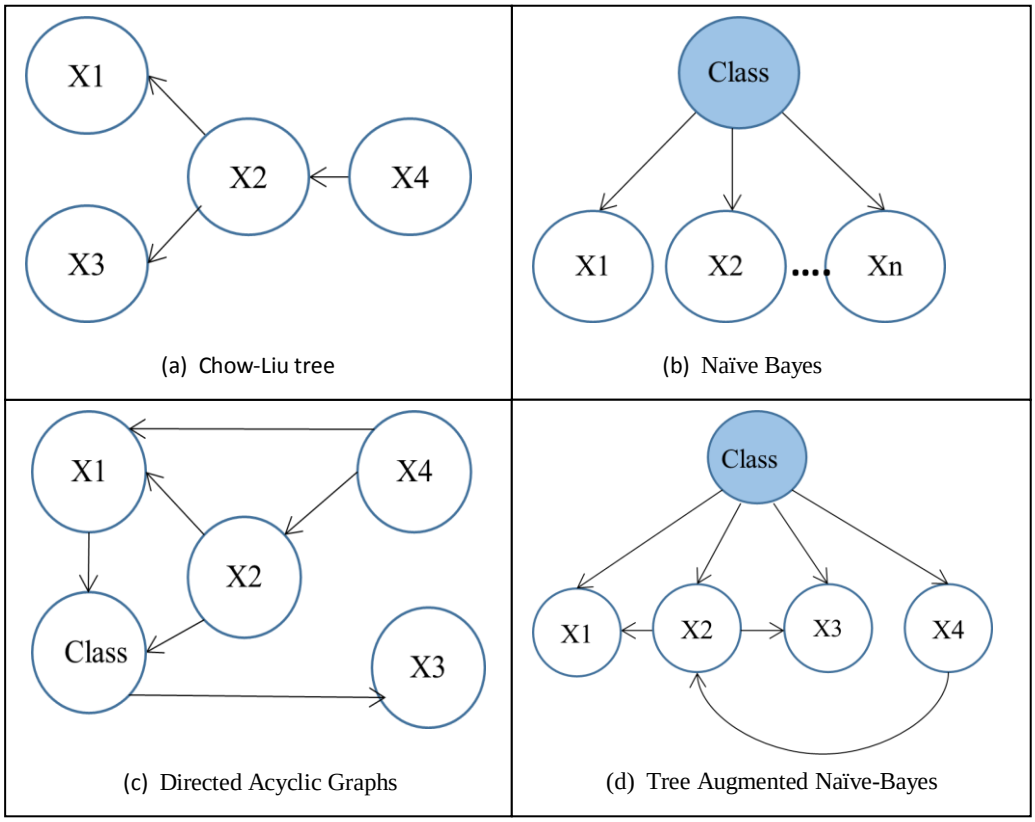


Figure 2.8: Bayesian network structures.

Historically, many Bayesian network structure learning algorithms have been developed, these algorithms generally fall into three approaches (Cheng and Greiner 1999):

- *Scoring-and-search-based approach:* find the BNs that maximizes score (Cooper and Herskovits,1992; Heckerman et al., 1995, Chickering, 2002).
- *Constrain-based approach (CI-based approach):* it called also Conditional Independent based algorithms, where it based on data by selecting for each variable a set of candidate parents (Spirtes et al., 1993; Cheng et al., 1997).
- *Hybrid approach*: that combines both of these approaches together to learn a BN structure.

Figure 2.9 presents a diagram that shows some references under each category and is followed by a description of the main categories (Carvalho, 2009; Cheng and Greiner, 1999).

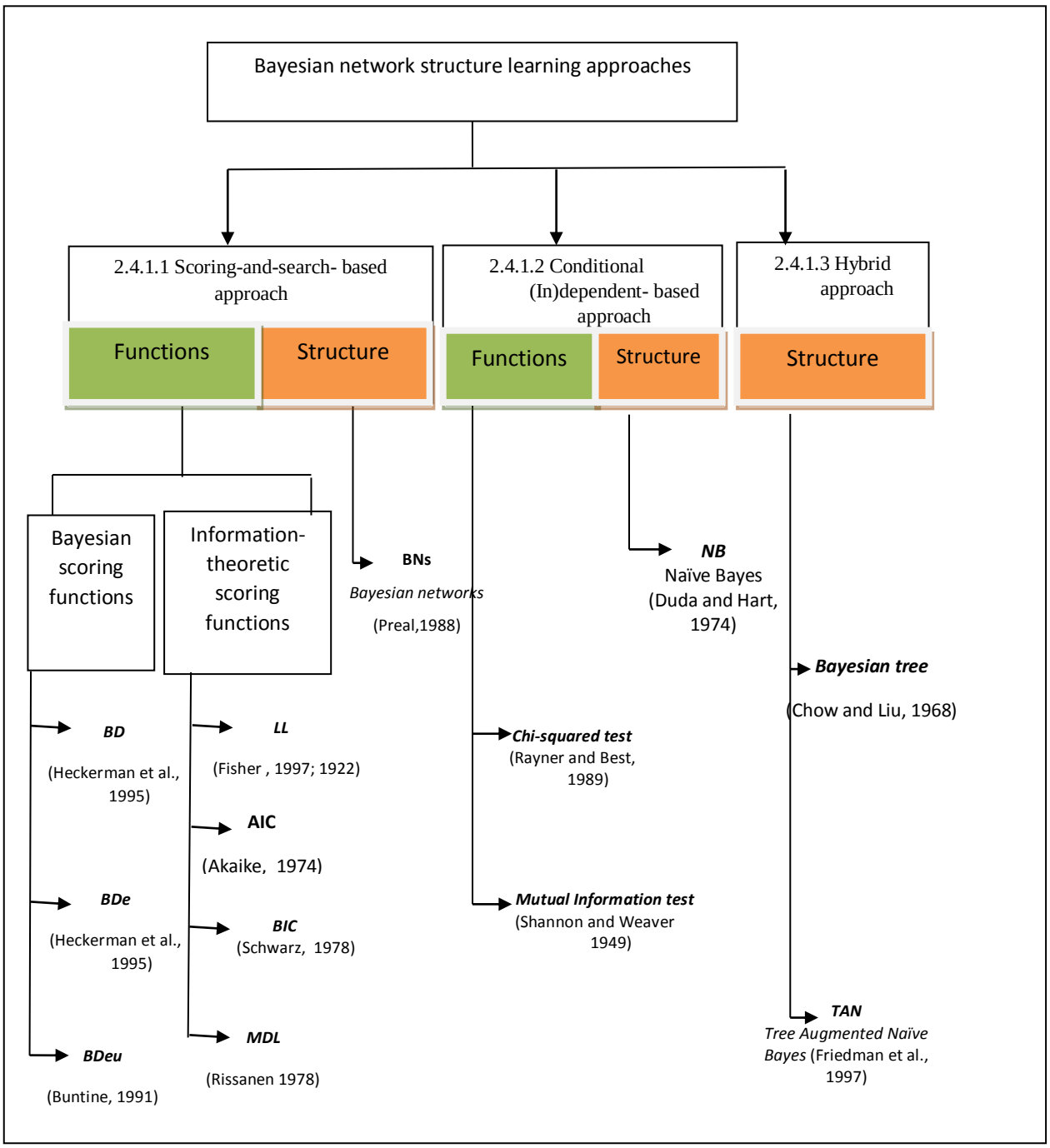


Figure 2.9: Bayesian network structure learning approaches.

2.4.1.1 Scoring-and-search-based approach

The task of finding a structure of a Bayesian network that describes the observed data is difficult and time-consuming, and has been shown to be an *NP-complete problem* (Chickering1996, 2004). Practically, when the search space is extremely large, the search

procedure will spend a lot of time examining unreasonable candidate structures, where the search space represents all the possible BNs structures. For example, Table 2.1 shows all possible structures of directed acyclic graphs (DAGs) given the number of variables (nodes) in the domain. Thus, when the number of nodes are large, then the number of possible DAGs are extremely large. Robinson (1973, 1977) derived the following efficiently computable recursive function to determine the number of possible structures that contain n nodes:

$$f(n) = \sum_{i=1}^{n}(-1)^{i+1} C_i^n 2^{i(n-i)} f(n-i) \qquad (2.7)$$

Where, n represents the number of variables, and C_i^n is $=\binom{n}{i} = \frac{n!}{i!\ (n-r)!}$

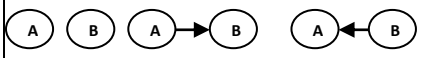
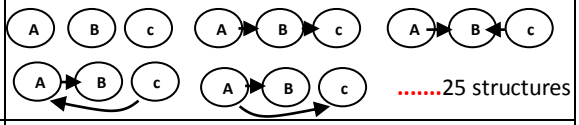
Number of variables in structure	Number of the possible BN structures	All possible BN structures
1	1	
2	3	
3	25	 .....25 structures
4	543	.
5	29,281	.
6	3,781,503	.
7	1,138,779,265	.
8	78,370,2329,343	.
9	1,213,442,454,842,881	.
10	4,175,098,976,430,598,100	

Table 2.2: Number of BN structures based on number of nodes (Laskey, 2015).

Cooper (1990) argued that given this is an NP-hard problem, we need to find "*approximate solutions*". The first attempts at finding approximate solutions were by Chow-Liu (1968) who developed branching algorithms to learn *Bayesian trees.* Dagum and Luby (1993) showed that even finding approximate solutions is NP-hard, thus they introduced a new method that restricted the possible parents of each node. After that, Dasgupta (1999) introduced *2-polytrees* (a singly connected network) which is also NP-hard. Finally, *heuristic search methods* have been proposed for addressing the problem of learning BNs in polynomial-time.

The scoring-and-search-based approach uses heuristic search algorithms to learn Bayesian network structures with respect to a goodness of fit score (Cheng and Greiner 1999).

Heuristic search methods are based on two steps:

- **Using *search methods* to build the structure**: fundamentally, there are several types of search algorithms such as *greedy hill climbing, simulated annealing, Genetic algorithm, Tabu search, best first search, K2 algorithm,* etc (Cooper and Herskovits, 1992). Most learning algorithms employ different search methods but the same search space. However, each search algorithm is based on a set of *search operators*; these operators are used to transfer a BN structure from one state to another state, such as *arc addition*, *arc deletion*, and *arc reversion*. As shown in Figure 2.10, starting from an initial network structure, one can apply the search operators (without introducing a cycle) to create the set of candidate neighbouring structures. A scoring or evaluation function can then be used to aid the selection of the next state as part of the search process, then the structure that has the highest score is selected (Vandel et al., 2012).

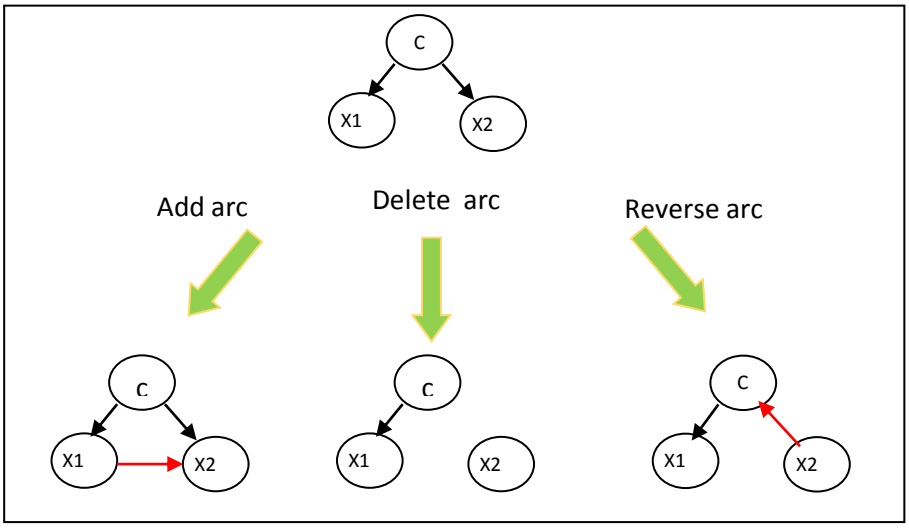


Figure 2.10: Set of operators (Vandel et al., 2012).

- **Using *scoring functions* to evaluate each structure**: score functions use to aid the search process to evaluate the structure. The scoring-and-search based approach starts from an initial random structure and moves to its neighbours by using the transition operators (as illustrated in Figure 2.10) to suggest new structures. The scoring function is used as an evaluation function and the search continued until no further improvement can be obtained. Figure 2.11 illustrated the idea where there are two nodes and a link is added resulting in an improved score.

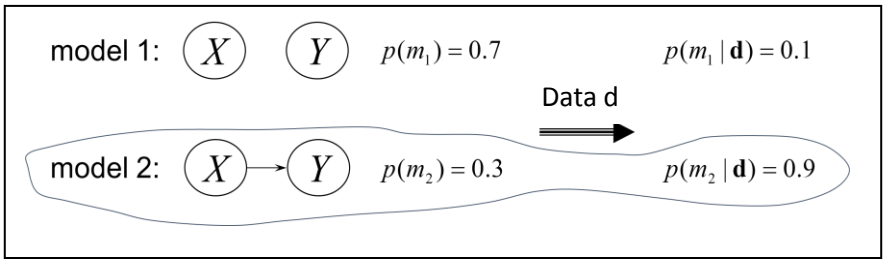


Figure 2.11: Model selection that maximize the score given data (Meek, 2015)

As shown in Figure 2.9, scoring functions are divided into two groups: *Bayesian scoring functions* and *information-theoretic scoring functions* (Heckerman et al., 1995), which are described below.

i. Bayesian Scoring functions are based on calculating the posterior probability using Bayes theorem and include two functions, both based on Bayesian Dirichlet (BD) functions (Heckerman et al., 1995). These functions are *BDe* where 'e' is for likelihood-equivalence (Heckerman, et al., 1995) and *BDeu* where 'u' denotes uniform joint distribution (Buntine, 1991).

ii. Information-theoretic scoring functions are based on the view that the best models are those that are the most succinct at representing the data, where the data is compressed into a shorter message length. Two common measures are the *Log Likelihood* (LL) score (Fisher, 1997; 1922) and the *Minimum Description Length* (MDL) (Rissanen, 1978), both of which have been shown to be effective in a number of studies (Friedman, 1997) and described in more detail below.

- **The Log Likelihood (LL) score**

  Several authors have described how the log likelihood measure can be used to assess the extent to which a given Bayesian network that represents data distribution. The following description is taken from Grossman and Domingos (2004) to analyse the LL score function. Consider a training set D=$\{X_1, \ldots, X_n\}$, the goal is to find the Bayesian network B that best representation the joint distribution P(X|Θ) where Θ are parameters where,

the likelihood of having parameters Θ given the data X_i is defined by (Grossman and Domingos, 2004) as:

$$L(\Theta|X_1, \dots, X_n) = \prod_{i=1}^{n} P(X_i|\Theta)$$

Then, applying the *natural log function*, because, logs reduce potential for underflow in numerical analysis, due to very small likelihoods.

$$\log \quad L(\Theta|X_1, \dots, X_n) = \sum_{i=1}^{n} \log P(X_i|\Theta)$$

From which the maximum likelihood estimator $\hat{\Theta}_{MLE}$ is defined as:

$$\hat{\Theta}_{MLE} = arg_{\Theta} \ max \sum_{i=1}^{n} log \, P(X_i|\Theta)$$

In particular, choosing the parameter value that makes the data actually observed as likely as possible.

$$LL(\Theta|D) = \sum_{i=1}^{n} \log P(X_i|\Theta)$$

The log-likelihood in BNs of n nodes, and m values of each node can be expressed in the following way (Campos, 2006):

$$LL(B|D) = LL(X_i|X_j) = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{ij} * \log\left(\frac{N_{ij}}{N_j}\right) \qquad (2.8)$$

The log-likelihood function when node X_i takes its parent X_j is shown in equation (2.8), where N_{ij} is the number of instances in the data D that has the intersection between node values i, and j, and N_j is the number of all instances in data D that has j value. As an example, consider the simple Bayesian network to explain the concept of LL score function is shown in the Appendix B1.

The LL score can quickly learn complete network structures, but it cannot provide a useful representation of the independence assumptions of the learned network (Campos, 2006). That is, this score is extremely specific but it cannot give a good structure if the model is over trained, obviously, the model becomes too specific because it adds too many links. Therefore, several theoretic scoring functions have been introduced to

devolp LL and avoid overfitting by limiting the number of parents per network variable, and by using some penalization factor over the LL score, such as the MDL function described below.

- **Minimum Description Length (MDL)**

The Minimum Description Length score (MDL) (Rissanen, 1978) is a formalization of Occam's razor**:**

"The best hypothesis for a given set of data is the one that leads to the best compression of the data."

Rissanen (1978) introduced the MDL score and his idea was based on how to reduce each model to bits. He stated that if the sender takes a set of observations dataset as input, then encodes these observations and sends a message that contains all the information about the model to a receiver, the receiver should be able to decode the message and produce the original message using the model. A good model will be one that is of minimal length. More precisely, suppose that: D is a set of observations dataset, B a Bayesian model that is used to describe D, L(B) represents the length of the code in bits necessary to encode the model B, and L(D|B) represents the length of the data D encoded using the Bayesian model B (Ramos, 2006). Where, the total length of the message is presented in equation (2.9), which includes the length required to represent the network L(B) plus the length necessary to represent the data given the network L(D|B) (Friedman and Goldszmidt, 1998).

$$L = L(D|B) + L(B) \qquad (2.9)$$

In particular, the first part L(D|B) is the log likelihood score function LL(B|D)that described in equation (2.8), where it represents how many bits are needed to describe D when encoded with B. While, the second part of equation (2.9), namely L(B), represents the number of bits used to represent and encode the model B and its parameters Θ . It called penalization factor, can be expressed in the following way (Campos, 2006):

$$L(B) = \frac{\log N}{2}|\Theta| \qquad (2.10)$$

Where $|\Theta|$ represents the number of parameters Θ in the network B, and N is the total number of instances in data D. In particular, when L(B) is 0, then the MDL will be equal

to LL score. Figure 2.12 illustrates this for Bayesian networks in which the first part represents the log likelihood function, and the second part represents proportionality factor of MDL score that shows in equation (2.10).

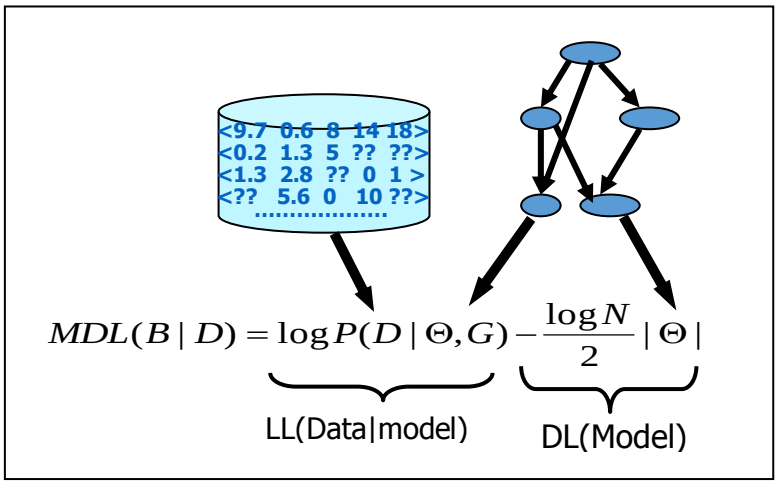


Figure 2.12: Illustration of the concept of data compression in MDL (Rish, 2015).

The MDL scoring function of a network *B* given a training dataset *D*, is written as MDL(*B*|*D*) (Friedman, 1997; Neapolitan, 2004),is given by:

$$MDL(B|D) = LL(X_i|X_j) = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{ij} * \log(\frac{N_{ij}}{N_i}) - \frac{\log N}{2}|\Theta|$$

$$MDL(B|D) = LL(B|D) - \frac{\log N}{2}|\Theta| \qquad (2.11)$$

The literature also contains two variations of the MDL score:

- **The Akaike Information Criterion (AIC),** (Akaike, 1974), where the penalization factor = 2 |Θ| as :

$$AIC\ (B|D) = 2\ LL(B|D) - 2\ |\Theta| \qquad (2.12)$$

- **The Bayesian Information Criterion (BIC),** (Schwarz, 1978) which takes the form:

$$BIC(B|D) = 2\ LL(B|D) - \log N\ |\Theta| \qquad (2.13)$$

All of the above score functions have different characteristics (Friedman and Goldszmidt, 1998; Campos, 2006) which can be summarised as follows:

In particular, the LL score function is not suitable for learning the structure of Bayesian networks, because it requires an exponential number of parameters, and that will lead to have a high variance, and poor prediction (overfitting problem). To address this problem, the AIC, BIC, and MDL measures use some penalization factor over the LL score. According to Maimon and Rokach(2005), AIC score penalises the LL(B|D) with a term that increases linearly with the number of parameters $|\Theta|$ of the model B. However, the AIC score does not lead to a consistent estimation when the model is unknown(Maimon and Rokach, 2005), because it is based on the implicit assumption that $|\Theta|$ remains constant when the size of the example increases as shown in equation (2.12), obviously, it does not include the number of examples N. In contrast, the BIC measure includes the number of examples as shows in equation (2.13), though this can also lead to problems when N is large, since the variance term in the mean squared error expression will be negligible (Maimon and Rokach, 2005). On the other hand, the MDL score aims to resolve this problem, and according to (Friedman, 1997), MDL avoids overfitting the data, by regulating the number of parameters learned and results in learning a structure that reflects the distribution better.

All of the above score functions can be used on any Bayesian network structures such as DAG, CL tree, TAN,... etc, to find high scoring structures for a given dataset D. (Cooper and Herskovits, 1992; Heckerman,1997).

2.4.1.2 Conditional independent-based approach

This approach is also called the *constraint based* approach. It selects for each variable a set of candidate parents and encodes a group of conditional independent relationships among them, according to the concept of *d-separation* (Pearl, 1988) which assess whether two variables are independent given other variables (see Appendix A for further details). This approach uses statistical tests functions such as *chi-squared test* (x^2 test) (Rayner and Best, 1989; Zibran, 2007), *mutual information test* (Shannon and Weaver, 1949; Cover and Thomas 2012), these tests use to find the conditional independence relationships among the attributes and uses these relationships as constraints to construct a BN. The Conditional Independent-based approach can lead to a simple Naïve Bayes structure of the kind described in Section 2.4.1 and illustrated in Figure 2.8(b).

2.4.1.3 Hybrid approach

This approach combine both of score-search approach and constraint approach together to learn the structure of a BN. Two such algorithms include learning as Chow-Liu tree (Chow and Liu, 1968), and Tree Augmented Naïve-Bayes networks TANs (Friedman et al.,1997).

.

2.4.1.3.1 Chow-Liu tree

Chow and Liu (1968) describe a procedure for constructing a Bayesian tree from data (also called a CL tree). The procedure constructs an approximation of the Bayesian network using information function, where the original algorithm used *Mutual Information (*MI) function, but it can be used on any score functions or conditional independent function thus, this algorithm is hybrid. In particular, it uses only O(N^2) pair wise dependency calculations, where *N* is the number of nodes (Cheng and Greiner 1999). The CL algorithm can be summarised in five steps (Friedman et al., 1997):

- ***Step 1:Compute Mutual Information:***

Consider a graph G = (V, E), let V denotes a set of discrete random variables, V={$X_1, X_2, X_3, \ldots, X_n$}, where E is a set of edges. First the marginal distributions of both $P(X_i, X_j) = \frac{Nij}{N}$ and P (X_i)= $\frac{Nij}{N}$ are computed from the data, where i, j belong to V. Then, use these marginals to compute the mutual information values of all n(n-1)/2 pairwise mutual information gains $MI(X_i, X_j)$, where i={1,2,3,...,n-1}, and j={i+1,,...,n}and i<j. Mutual Information calculated as shown in equation (2.14).

$$MI(X_i, X_j) = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} P(X_i, X_j) \ * \ \log \frac{P(X_i, X_j)}{P(Xi)P(Xj)} \ , \text{where i} \neq \text{j} \qquad (2.14)$$

- ***Step2: Build a complete undirected graph:***

A complete undirected graph is then built, where the edges between X_i, and X_j are set to a weight corresponding to the mutual information $MI(X_i, X_j)$.

- ***Step3: Apply (MWST) algorithm:***

A maximal spanning tree is then obtained using a Maximum Weight Spanning Tree (MWST) algorithm (Cormen et al., 1990). *Maximum weight dependence tree* is constructed branch by branch as shown in example in Figure 2.13, where it uses

$MI(X_i,X_j)$ as the weight for edge (X_i,X_j), for all $i,j \in V$, and $i \neq j$.

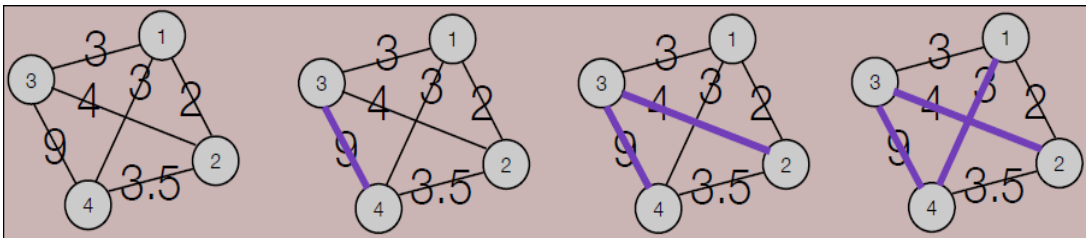


Figure 2.13: How MWST finds a tree with the greatest total weight (Hong, 2007).

***Step4: Convert to directed tree**:*

Convert undirected tree into a directed tree by choosing any node a root node and setting the directions of the links to be outward from it.

Where, as simple CL tree shows in Figure 2.8(a), and a real example in Appendix B shows how this procedure works, Figure B1.7 represents a simple Chow-Liu tree (directed tree).

2.4.1.3.2 Tree Augmented Naïve-Bayes (TAN) structure

The TAN classifier was introduced by Friedman et al. (1997), as an extension of Naïve Bayes networks by allowing the attributes to form a tree to represent the dependencies among the attributes, and relaxing the independence assumptions (Cheng and Greiner 2001). Figure 2.8(d) shows a TAN structure where each node has two parents, a class node and another node. A TAN is unlike a Naïve Bayes network since it can model all dependencies between variables; in particular, it is a less restricted structure than Naïve Bayes by allowing one parent per variable in addition to the class. A TAN is formed by calculating the maximum weight spanning tree algorithm using the Chow-Liu algorithm. Friedman et al.(1997) argue that the drawback of using LL score function, stems from that it does not work well when the number of instances is limited, as was described previously in Section 2.4.1.1. Thus, Friedman et al. (1997) suggested the use of a restricted log likelihood function that is called *conditional log likelihood* (CLL) function (Spiegelhalter et al., 1993). Their suggestions were based on the assumption that by maximising the conditional log likelihood, it is possible to learn a model that best approximates the conditional probability of class C given the attribute values. Therefore, Friedman et al. (1997) conclude that the model that maximises this CLL

function yields the best classifier. Friedman et al. (1997) developed the following algorithm for learning TANs based on the MDL as score function instead of MI function:

Step 1*: *Compute Conditional Information:

Consider a graph G = (V, E), let V denotes a set of discrete random variables, V={$X_1, X_2, X_3, \ldots, X_n, C$}, where *E* is a set of edges, and the edges (X_i, X_j) between tree represent the weight of MDL between (X_i, X_j) based on class node C. In particular, the first step is applying MDL that based on conditional log likelihood CLL score function to obtained a maximum weight dependence tree. Where the weight between nodes (X_i, X_j) represents the difference between *MDL dependent nodes* as MDL($X_i|X_j, C$), and *MDL dependent class* as MDL($X_i|C$), as shown in the Appendix B example B1.3. *MDL dependent nodes*, and *MDL dependent class* are calculated as shown in equations (2.15), and (2.16) respectively.

$$MDL(X_i|X_j, C) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1} N_{ijk}\ log\ (\frac{N_{ijk}}{N_{jk}}) - \frac{\log N}{2}|\Theta| \quad (2.15)$$

$$MDL(X_i|\ C) = \sum_{i=1}^{n}\sum_{k=1} N_{ik}\ log\ (\frac{N_{ik}}{N_k}) - \frac{\log N}{2}|\Theta| \qquad (2.16)$$

Where N_{ijk} is the number of instances in the data D that has the intersection between node values i, j, and class k. While, N_{jk} is the number of all instances in data D that has the intersection between node j, and class node k. Also, N_{ik} is the number of instances in the data D that has the intersection between node i, and class node k. While, N_k is the number of all instances in data D of class k. Where, Θ are parameters, and N number of all instances in the data D.

Step2: Build a complete undirected graph:

Construct an undirectedcomplete graph between all the attributes (excluding class variable), where the edge weight is calculated from previous step.

Step3: Apply (MWST) algorithm:

Build a maximum weighted spanning tree by running a Maximum Weight Spanning Tree (MWST) algorithm (Cormen et al., 1990).

Step4: Convert to directed tree*:*

Convert the resulting undirected tree to a directed tree by choosing a root node and setting the direction of all edges to be outward from it.

Step5: Add the class label as root:

Construct a TAN model by adding a class label node as root for dependency tree; adding an arc from *C* to all X_i.

Clear example will be show in Appendix B1, to illustrate how to learn TAN structure based on the algorithm steps. In comparison to the other approaches described above, Friedman et al. (1997) point out that:

- Learning TANs involves no process of searching.
- TANs are more robust than Naïve Bayes; because they are based on relaxing the independence assumptions.
- TAN algorithm can be learned in polynomial time; thus, it is faster than other BNs.
- TAN classifier is more accurate; because it is based on maximising the restricted weight between nodes that yields to an improved classification process.
- Based on the experiments of Friedman et al.(1997), the learning procedures of TAN are guaranteed to find the optimal tree structure.

2.4.2 Bayesian network parameter learning

After learning the structure of a Bayesian network, the next step aims to *learn the parameters;* that is the conditional probabilities between nodes and their parents which can be viewed as a *Conditional Probability Tables* (CPTs). A conditional probability table represents the dependency between variables. For example, Figure 2.14 shows a dependency between two nodes ‘cancer disease’ and ‘test’, with the prior probability of ‘having cancer’ being present is 5% and ‘not having cancer’ being absent is 95%. The extent of the

dependency is quantified by the CPT, which for example, indicates that the probability of test being positive is 75% if cancer is present.

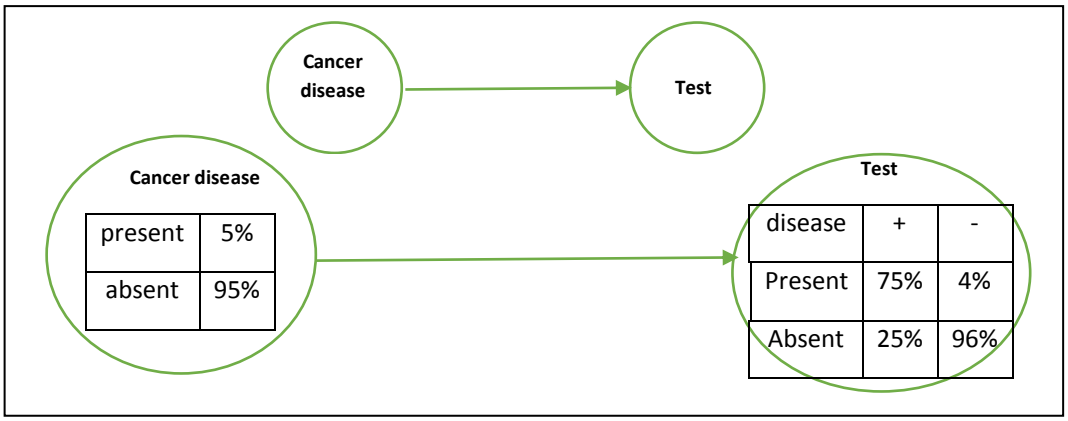


Figure 2.14: A simple BNs model with CPTs.

To obtain these probabilities, a *Simple Estimator* (SE) is used and takes the form given by equation (2.17) (Bouckaert, 2004).

$$P(X_i|X_j) = \frac{N_{ij} + \alpha}{N + (\alpha * n_{X_i})} \qquad (2.17)$$

Where, X_j is parent of X_i after learn structure, N_{ij} represents the number of the events X_i, and parent node X_j occurring together in the data; N is the total number of examples of parent node X_j. While, n_{X_i} is the number of values of node X_i. Where, $\alpha = 0.5$ represents the initial count on each value to avoid 0. For example, to illustrate the Simple Estimator SE, suppose we have Table 2.3 from play-tennis dataset that represents as:

Outlook	Wind
Sunny	FALSE
Sunny	TRUE
overcast	TRUE
rainy	TRUE
rainy	TRUE
rainy	TRUE
overcast	TRUE
sunny	FALSE
rainy	FALSE

Table 2.3: A simple play-tennis dataset with two attributes.

Then the simple estimator estimates of the probability it is sunny and windy which happened one time is: $P(\text{Outlook} = \text{'Sunny'} \mid \text{Wind} = \text{'True'}) = \frac{1+0.5}{6+(3*0.5)} = 0.2$

With the same way the simple estimator is used to estimate all the attribute values between the nodes in the structure and save the values into CPTs as shown in Figure 2.15.

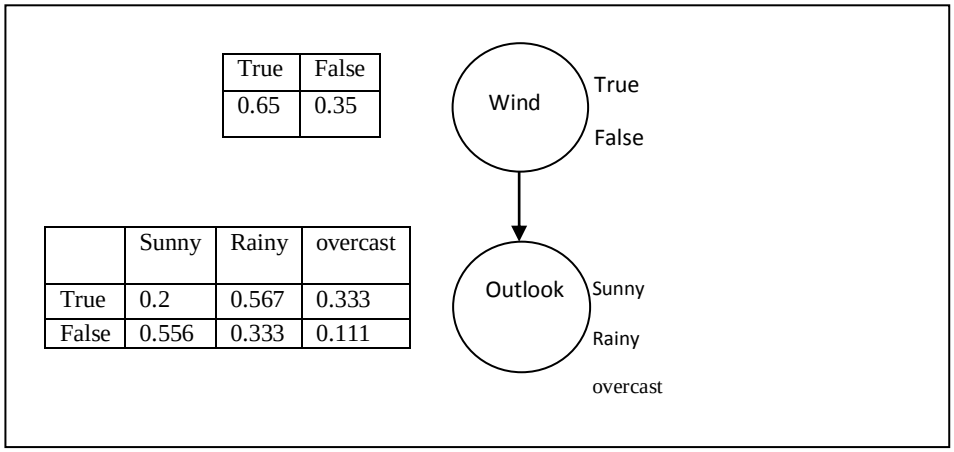


Figure 2.15: A simple network structure for the play-tennis dataset and the associated CPTs.

After learning the Bayesian network from a dataset (structure, and parameters), it can be used as a classifier to classify new instances, as shown in Appendix B, in Figure B1.8, where real dataset has been applied on WEKA software to show how TAN structure learned from play-tennis dataset and how it learned parameters, then how it used as classifier.

2.5 Summary

In summary, this chapter has presented the background on Bayesian networks. It described the data classification processes, and the types of classification algorithms such as decision trees, neural networks, and Bayesian networks classification algorithms. Then, it presented the principles of Bayesian network algorithms that are based on probability theory. Moreover, it outlined inference in Bayesian networks, and how to update the probabilities of nodes given evidence. An illustrative examples were given to demonstrate type of inference. In addition, the chapter has discussed how to learn Bayesian networks (structure and parameters). It presents three approaches that are based on some functions to learn Bayesian structures: Scoring-and-search-based approach, constrain-based approach, and Hybrid approach. Furthermore, this chapter presented the main Bayesian structures that are used in this research which are Chow-Liu tree, and Tree Augmented Naïve-Bayes.

The following chapter presents a survey of existing cost-sensitive algorithms, and discusses the differences between existing cost-sensitive algorithm.

Chapter 3: Survey of Existing Cost-Sensitive Algorithms

This chapter presents a survey of approaches to cost-sensitive learning. As mentioned in Chapter 1, most of the existing research on cost-sensitive learning has focussed on decision tree learning and the aim of this chapter is to describe the strategies adopted with a view to adopting them for developing algorithms for learning cost-sensitive Bayesian networks. Section 3.1 describes cost-insensitive algorithms, Section 3.2 describes existing cost-sensitive algorithms. Section 3.3 describes the literature review in cost-sensitive Bayesian network algorithms. Section 3.4 presents a brief summary of this chapter.

3.1 Cost-insensitive learning algorithms

The classification task aims to distinguish instances in a dataset into known categories, called classes, in accordance to specific attribute values. The induction of classifiers from datasets of pre-classified instances is perceived to be a major challenge (Friedman, 1998). Thus, many methods and algorithms have been introduced as classifiers such as: decision trees, Bayesian networks, and neural networks. Most of these early machine learning algorithms, focused on maximizing accuracy, and assumed that costs for misclassification error remain equal (Mitchell, 1997). Early machine learning algorithms, now termed *Cost-insensitive learning algorithms*, focused on maximizing accuracy but did not take any type of costs into account (Mitchell, 1997). The measure *accuracy* is defined as given in equation (3.1) and denotes the proportion of correctly classified instances.

Several authors have noted that cost-insensitive learning is not adequate for practical applications (Turney, 2000; Vadera and Nechab, 1995; Domingos, 1999). For example, in medical diagnosis applications, the cost of a false positive (FP) includes unnecessary treatment and unnecessary worry while the cost of false negative (FN) error includes postponed treatment or failure to treat; and death or injury (Santos-Rodríguez et al., 2009). In fraud detection applications, a false positive (FP) error can lead to resources being wasted investigating non-frauds and reducing the benefits; while a false negative (FN) error such as a failure to detect fraud could be very expensive (Phua et al., 2006).

Hence, in recent years, a significant level of attention has been paid to *cost-sensitive learning algorithms*, including making accuracy-based learners cost-sensitive (Lomax and Vadera, 2013). Thus, many cost sensitive approaches are designed to reduce the cost of misclassifications rather than the number of misclassified examples.

3.2 Cost-sensitive learning algorithms

Cost-sensitive learning algorithms take costs into consideration and aim to minimize costs (Elkan, 2001). In particular, there is a cost involved in the learning process, where, the cost is very important in the classification process. The word cost is used to describe the term in a very abstract sense, where cost has different measurement units, such as monetary units (dollars), temporal units (seconds), or abstract units of utility (Turney, 2000). Cost should not only be a physical entity that could be measured, as cost sometimes includes time wasted, and loss of a patient's life, such as misclassifying a patient with cancer as having no cancer. As well as misclassification costs, Turney (1995) points out that test costs are also an important consideration. For example, in medical diagnosis applications, a blood test has a cost, so if the misclassification cost of diagnosis a patient is £10 and test cost is £2; so the misclassification cost is greater than the test cost, in this case, it is worthwhile to pay test costs because that seem to have some predictive value. On the other hand, if misclassification costs less than test costs, then there is no point in doing test costs.

The following example illustrates the use of a *cost-matrix* to calculate accuracy and costs. Table 3.1 presents an example cost-matrix C, where $C(i,j)$ is the cost of predicting an example to be in class i when it is actually in class j.

Predicting class	Actual class	
	Actual Positive	Actual Negative
Predicting Positive	TP=0	FP=£1
Predicting Negative	FN=£50	TN=0

Table 3.1: A cost matrix for two-class problems

A classification scheme, when applied to some data, will lead to outcomes that are correct or incorrect instances, resulting in what is known as a *confusion matrix*. For example, suppose we have two different classifiers, a decision tree classifier and a Bayesian network classifier. Applying these classifiers to the *Breast cancer* dataset and evaluating the supplied test set in the models may give the confusion matrixes in Table 3.2 (a) and (b) respectively.

Predicting class	Actual class	
	Actual no cancer	Actual cancer
no cancer	193	8
cancer	62	23

Table 3.2: (a): Outcomes from decision tree classifier (J48) on the Breast cancer dataset.

Predicting class	Actual class	
	Actual no cancer	Actual cancer
no cancer	173	28
cancer	59	26

Table 3.2: (b) Outcomes from Bayesian network classifier (TAN) on the Breast cancer dataset.

Given the outcomes in Tables 3.2(a) and 3.2(b), we can compute the accuracy and misclassification costs of the two classifiers as using the following measures:

$$Accuracy = \frac{No.of\ correct\ examples}{Total\ number\ of\ examples} = \frac{TP + TN}{TP + FP + TN + FN} \quad (3.1)$$

$$Cost = \sum_{i=1}^{k} No.of\ missclassified_i \ * \ Cost\ (i,j) \quad (3.2)$$

Where, k is the number of classes, $No.of\ missclassified_i$ is the number of class i examples that are misclassified, and $Cost(i,j)$ is the cost of misclassifying examples of class i when j is given. Using these equations, we obtain the following accuracies and costs for the decision tree classifier (DT) and the Bayesian network classifier (BN):

DT Accuracy $=\frac{193+23}{286}$ =75.52%, and DT misclassification costs=£50*62 +£1*8 = £3108

BN Accuracy $=\frac{173+26}{286}$ =69.58%, and BN misclassification costs=£50*59 +£1*28 = £2978

Thus, in this example, applying the Bayesian network classifier will entail less costs than applying the decision tree classifier on the Breast cancer dataset.

3.2.1 Cost sensitive algorithms categories

As illustrated by the previous example in Section 3.2, a good cost-sensitive classifier should be able to predict the class of an example that leads to the lowest expected cost (Elkan, 2001),

where the expectation is computed after applying the classifier by using the expected cost function (Zadrozny and Elkan,2001), as given in equation (3.3) . Assume that (i, j) represents 2 classes in cost matrix C, if i=j then the prediction is correct, while if i≠ j the prediction is incorrect. The expected cost of classifying an instance x into true class i, can be expressed as:

$$expected\ cost(x,i) = \sum_{j=1}^{k} P(j|x) * Cost(i,j) \qquad (3.3)$$

Where, k is the number of classes; $P(j|x)$ represents the probability estimation of classifying the instance x into class j; and $Cost(i,j)$ is the cost of misclassifying class i when j is given; obviously the cost of predicting x to class i when the true class of x is j.

Several authors have categorised cost-sensitive induction algorithms differently. According to Zadrozny et al. (2003b) cost-sensitive classifiers can be divided into two categories: *Black Box* and *Transparent Box*. Black box methods use a classifier as closed box without changing its behaviour and can work for any classifier. On the other hand, transparent box methods require knowledge of the particular learning algorithm and are based on changing the algorithm to include costs. Ling and Sheng (2010) use the terms *direct methods*, and *indirect methods*, where direct method includes cost directly during building a cost sensitive learning algorithm; introducing and utilizing misclassification costs into the learning algorithms. While, indirect method includes cost before or after applying the algorithm; by pre-processing the training data, or post-processing the output of a cost-insensitive learning algorithm. As well as these methods, a further category involves using *evolutionary algorithms*. The literature search identified several methods under these categories which are presented in Figure 3.1 and described below.

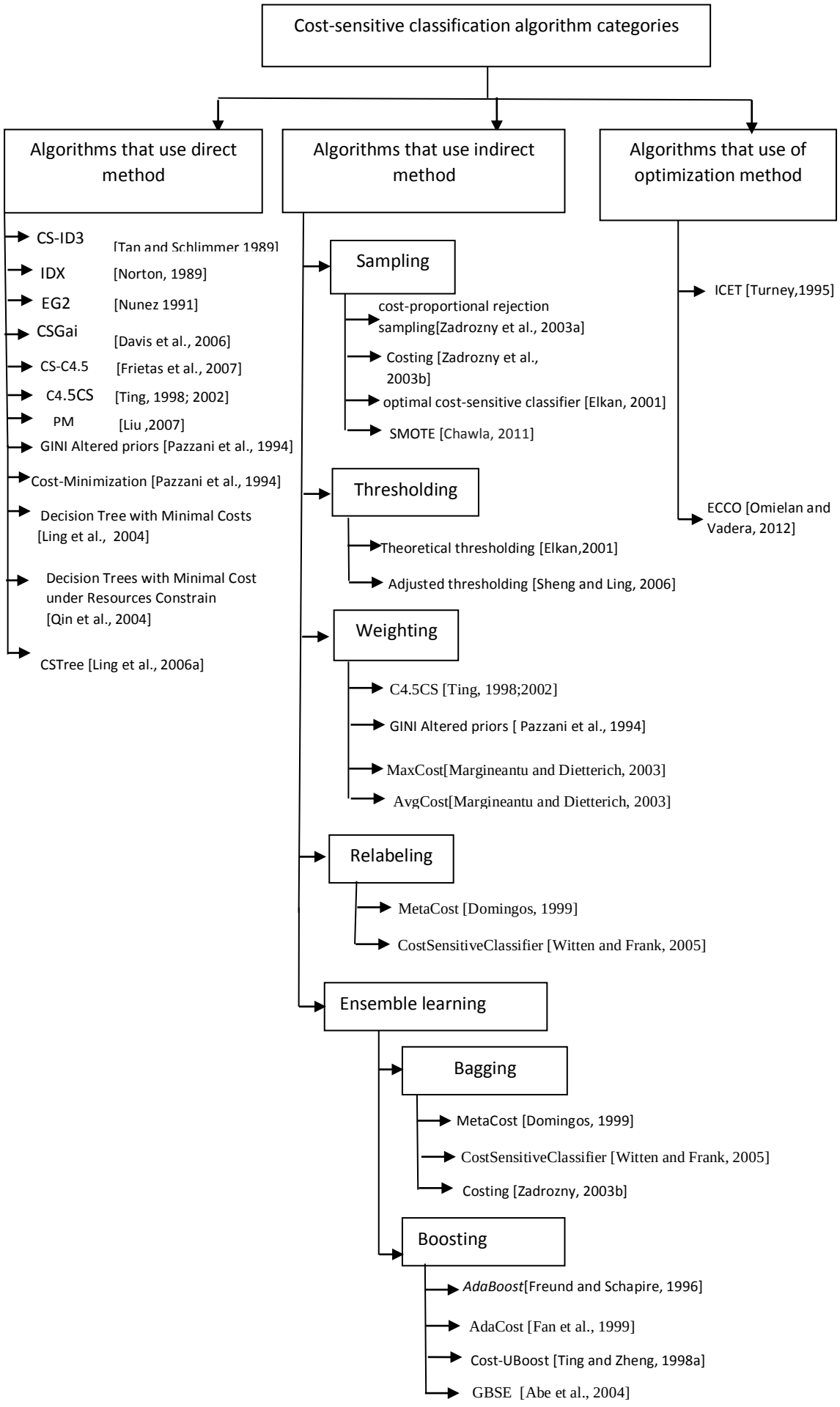


Figure 3.1: Cost-sensitive learning categories

.

3.2.1.1 Algorithms that use direct methods

A key step in decision tree learning algorithm is selecting the next attribute of the decision tree, which is done by using a measure (Quinlan, 1979) such as information gain that is based on computing the difference between entropy of classification before and after an attribute's value is known. The following equations define how this is computed for a class C and attribute A:

$$Entropy(C) = \sum_{c \in Class} -P(c) * \log_2 P(c)$$

$$Entropy(Att) = \sum_{a \in Att} P(a) * \sum_{c \in Class} -P(a|c) * \log_2 P(a|c)$$

$$InfoGain_A = Entropy(C) - Entropy(Att) \qquad (3.4)$$

Where, c is a class value, and a is an attribute value of A and $InfoGain_A$ is the information gain of the attribute A. The attribute that results in the highest information gain is used as the next attribute and the process repeated recursively until a stopping condition, such as a certain proportion of examples belonging to the same class is reached. However, this selection measure does not take account of costs. Hence, several algorithms have been introduced to include costs by *amending the statistical measurement*; or by *modifying and utilizing the cost directly* during the decision procedure (Lomax and Vadera, 2013).

i. Algorithms that amend the information theoretic measure

As mentioned in Section 3.2, the main two costs are: test costs and costs of misclassification. Test costs can be included by amending the selection measure to include the cost of a test. Algorithms using this approach include EG2 (Núñez, 1991), *CS-ID3* (Tan and Schlimmer, 1989), *IDX* (Norton 1989), and *CS*-C4.5 (Freitas et al., 2007). These algorithms adapt the information theoretic measure by developing a cost based attribute selection measure, called the *Information Cost Function for an attribute A* (ICF_A):

$$EG2: \qquad ICF_A = \frac{2^{InfoGain_A} - 1}{(Cost_A + 1)^{\omega}} \qquad (3.5)$$

$$CS-ID3: \qquad ICF_A = \frac{(InfoGain_A)^2}{Cost_A}$$

$$IDX: \qquad ICF_A = \frac{InfoGain_A}{Cost_A}$$

$$CS-C4.5: \qquad ICF_A = \frac{InfoGain_A}{(Cost_A\ \phi_A)^{\omega}}$$

All of these include the cost of attribute ($Cost_A$), they take account of the information gained. EG2 and CS-C4.5 also use a user provided parameter ω that varies the extent of the bias, while $\phi_A$in CS-C4.5 represents a risk factor of delayed tests; where there is a delay in the result of a test; for example, in a medical diagnosis a doctor sends a blood test to a laboratory, and the result might be delayed.

A natural way of amending such algorithms to take account of the cost of misclassifications is to modify equation (3.4) by altering the class probability P(i) so that it takes account of the relative costs of misclassification. In general:

$$\text{Probability}_i : P_i = \frac{N_i}{N} \qquad (3.6)$$

Where N_i is the number of examples of class i, and N is the total number of example, Breiman et al. (1984) introduced a method that modified this prior probability with *altered probability* as shown in equation (3.7) to take account of costs by weighting each prior P_i by the relative cost of misclassifying examples of class i; (Cost ratio_i).

$$\text{Altered Probability}_i = \text{Cost ratio}_i * \left(\frac{Ni}{N}\right) \qquad (3.7)$$

$$\text{Cost ratio}_i = \frac{\text{cost of missclassifiaction of class i}}{total\ missclassifaction\ costs} = \frac{\text{cost (i, j)}}{\sum_{j=1}^{k} \text{cost (i, j)}} \qquad (3.8)$$

Where, Cost ratio_i represents the cost ratio of class i; the cost proportion of class i to the total costs. For example, form cost matrix that is represented in Table 3.1 shows the cost ratio of positive class=50/51= 0.98, while, the cost ratio of negative class=1/51= 0.02. Also, Ni is the number of examples in class i, while N is the total number of examples.

As given in equation (3.7), this is the altered probability measure that can then be used in the information gain measure and the rest of the algorithm can remain unchanged. Ting (1998, 2002) also uses to modify the estimated probability of class i as shown in equation

(3.9). According to Pazzani et al. (1994) modified the estimated probability in *GINI index* measure (Breiman et al., 1984), and introduced new algorithm called *GINI Altered priors.*

$$\text{Altered GINI} = 1 - \sum_{i=1}^{k} (\text{Altered Probability }_i)^2 \qquad (3.9)$$

ii. Algorithms that utilize the cost directly

Instead of adapting the information gain to include costs, there are other algorithms which utilize the cost of misclassification and test costs directly as selection criteria. This category utilizes both costs during learning from the training data, where for each attribute in turn, the data is partitioned on that attribute's values. Then for each of the subsequent subsets created, the cost of errors is computed and then the sum of the costs of all these subsets is calculated to select the attribute that has minimum costs. Examples of algorithms that take this approach include *Cost-Minimization* (Pazzani et al., 1994), *Decision Tree with Minimal Costs* (Ling et al., 2004), *Decision Trees with Minimal Cost under Resources Constrain* (Qin et al., 2004), *CSTree* (Ling et al., 2006), and *PM* (Liu, 2007). For example in *Cost-Minimization* (Pazzani et al., 1994) without considering information gain, the attribute that results in the lowest misclassification costs is selected next (Pazzani et al.,1994).

3.2.1.2 Algorithms that use indirect methods

These methods include a cost as a separate stage in the learning process, and includes techniques such as *Sampling, Relabeling, Weighting, Thresholding,* and *meta methods (bagging* and *boosting)*. These methods can be applied before or after applying an existing accuracy based classifier and are described below.

3.2.1.2.1 Sampling

Sampling, also called stratification, it is used to amend the distribution of the data to reflect the costs of misclassification.The algorithms that are based on sampling, change the frequency of the data instances in the training set according to their costs. Sampling was used

to convert an insensitive cost learning process to sensitive cost learning by increasing the number of costly class examples or reducing the number of non-costly class examples to reflect their importance in cost sensitive learning process.

- ***Imbalanced data***

Imbalanced datasets occur where one class is rare while the other classes are frequent. It is often the case that the cost of misclassifying a rare example is significantly higher than a more frequent example (Suna et al., 2006). For instance, to detect a fraudulent customer, the cost of misclassifying a customer who commits fraud is greater than the cost of misclassifying a customer who is non-fraudulent. Figure 3.2 illustrates an imbalanced problem in two classes.

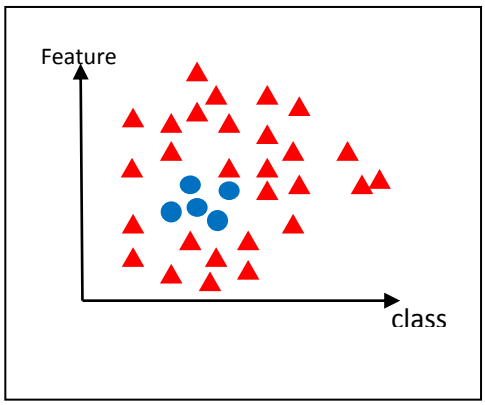


Figure 3.2: Imbalanced dataset .

In imbalanced datasets, building a classifier that does not consider the cost of misclassification, does not perform well because it is biased to classify most of the instances under the category of frequent class, which will result in producing misclassifying rare instances; obviously instances that belong to the rare class will be misclassified more than the ones belonging to the frequent class. Hence, sampling works with very highly skewed data (imbalanced data), because it aims to reduce the number of misclassification errors by using some mechanisms in order to provide a new data distribution that reflects misclassification cost (He and Garcia, 2009). However, several studies (Weiss and Provost, 2001; Laurikkala, 2001; Estabrooks, 2004) have found that a base classifier can improve its performance by balancing an imbalanced dataset (He and Garcia, 2009).

- ***Folk theorem:***

The key to using a sampling for cost-sensitive learning is a result known as the *Folk theorem* (Zadrozny et al., 2003a; Bailey and Elkan, 1994; Elkan, 2001). This theorem can be applied on any cost-insensitive classifier to turn it into a cost-sensitive classifier by changing the data distribution to reflect the costs. Zadrozny et al. (2003a) states that "*if the new examples are drawn from the old distribution, then optimal error rate classifiers for the new distribution are optimal cost minimizes for data drawn from original distribution"*. Formally, Zadrozny et al. (2003a) presents this change in the distribution as follows:

$$D'(\mathrm{x},\mathrm{y},\mathrm{c}) = \frac{\mathrm{C}}{\mathrm{E}_{\mathrm{x,y,c\sim D[c]}}}\mathrm{D}(\mathrm{x},\mathrm{y},\mathrm{c}) \qquad (3.10)$$

Where, the new distribution D' = factor * Old distribution D; x is the input space to a classifier; y is the binary that represents output space to a classifier; and C is the misclassification cost (Zadrozny et al., 2003a). Technically, the optimal error rate classifier from D' is the optimal minimizing cost from data which has been drawn from the original distribution D. Obviously, Zadrozny et al. (2003b) introduced a new sampling method based on the Folk theorem; they show that it is possible to change the distribution of the data to reflect the cost ratio. For example, consider a dataset, where the number of examples in class 1 is N1, and class 2 is N2, and the cost of misclassifying class 1 is C1, and class 2 is C2. Then, the new data distribution of N1 and N2 will be changed as shown in equation 3.11:

$$\frac{N_1'}{N_2'} = \frac{N_1 * C_1}{N_2 * C_2} \qquad (3.11)$$

Since, this theorem creates a new distribution from the old distribution by multiplying the old distribution with a factor proportional to the relative cost of each example the new distribution will be adapted with that cost. Therefore, this method makes a classifier get an expected cost minimization on the original distribution, and in the worst case this method can guarantee the classifier to give a good approximate cost minimization for any new sample.

In particular, there are several methods of sampling which correspond to all types of sampling, which are called *(a) Sampling-with-replacement*; it works by changing the data

distribution and taking random examples from a population, then returning these examples back into the population; where these examples can be selected more than one time, as shown in Figure 3.3(a). Hence, Zadrozy et al. (2003a) argue that using sampling with replacement can lead to overtraining because the duplication of examples, also all selected examples are not independent. On the other hand, *(b)Sampling without replacement* works by taking random examples from a population, then putting thes examples aside the population; where these examples can be selected one time, as shown in Figure 3.3(b). Hence, sampling without replacement insures that all examples in new distribution are drawn independently from old distribution, as a result, this type of sampling leads to an over-fitting problem (Zadrozny et al., 2003a).

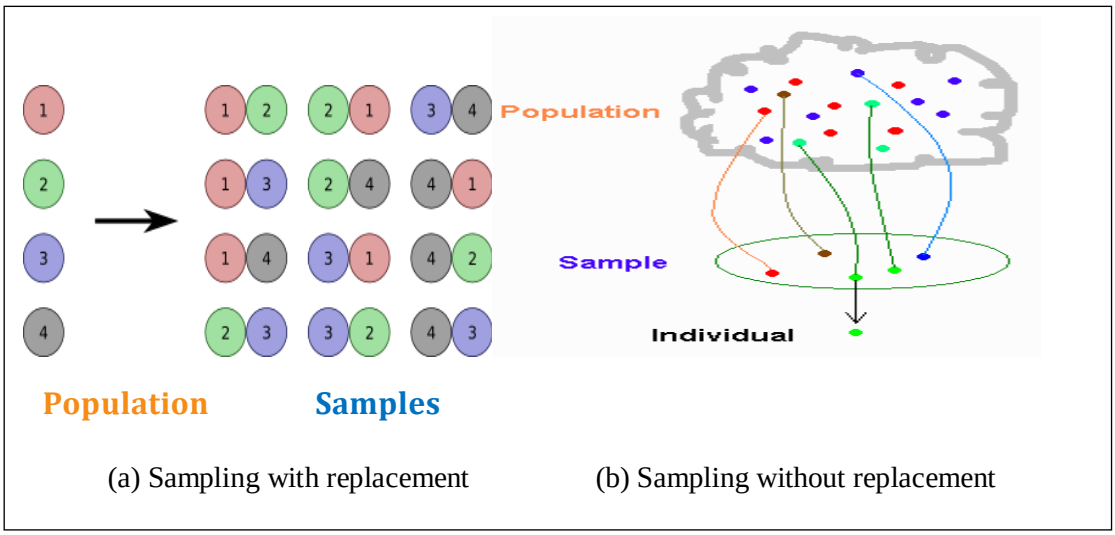


Figure 3.3: Sampling with / without replacement (WIKIbooks, 2015)

There are several methods for changing the distribution, including (Zadrozny et al., 2003a):

i. ***Over-sampling:*** this method of sampling increases and duplicates the number of rare class examples, without changing the frequent class examples. A potential problem with this method is over-fitting, because the minority class decision region becomes very specific, and will not be able to work accurately on the testing data. Also, increasing the number of training examples leads to increasing the learning time (Weiss et al., 2007).

ii. ***Under-sampling***: this method of sampling reduces the number of frequent class examples (the majority class), while keeping the original population of the minority

class. The problem with this method is that it can discard potentially useful instances, leading to misclassifying them (Weiss et al., 2007).

iii. Cost-proportionate rejection sampling: although, over and under-sampling methods give good results on some datasets, they do not work very well on others. Therefore, (Zadrozny et al., 2003b) proposed an alternative method based on rejection sampling, called *cost-proportional rejection sampling*. This sampling method avoids the over-fitting problem, and aims to minimize classification error. It works as the following steps:

- Drawing examples independently from the distribution as shown in equation (3.10).
- Then accepting the example with probability proportional to c/z, where z is chosen as the maximum cost of misclassifying an example, and c is the misclassification cost. Otherwise reject the example.
- Then using a learning classifier on the new distribution examples.

This sampling method will produce an approximately cost-minimizing classifier. In fact, the sample size of the new distribution is smaller than the original distribution because testing each example on that factor c/z will reject some examples. Hence, the time required for learning a classifier is much smaller (Zadrozny et al., 2003b).

iv. Cost-proportionate rejection sampling with aggregation (Costing): the *Costing* algorithm has been introduced by Zadrozny et al. (2003b), is based on different runs of cost-proportionate rejection sampling method described above, thus this method creates different training samples (distributions) in a very short time. Zadrozny et al. (2003b) utilize this feature to devise an ensemble learning algorithm (bagging) based on repeatedly performing cost-proportion rejection sampling from the original distribution *D* to produce multiple sample sets (new distributions)$\{D_1, D_2, D_3 \ldots D_m\}$. Figure 3.4 illustrates the *Costing* algorithm, which works as the following steps:

- Run cost-proportionate rejection sampling from the original sample (distribution) D, by accepting examples with probability c/z , then the new sample (distribution) will be created D_i, where, i=1 to m.
- Then, using a cost–insensitive classifier to learn a model from the new distribution D_i .

- Repeat the first, and the second steps *m* times, finally, getting several new samples ={D_1, D_2 , D_3...D_m}, and several models.
- The output classification is based on the average over all the models.

The goal in using averaging is to improve the performance of the classifier, and that gives approximate minimization of the classification error.

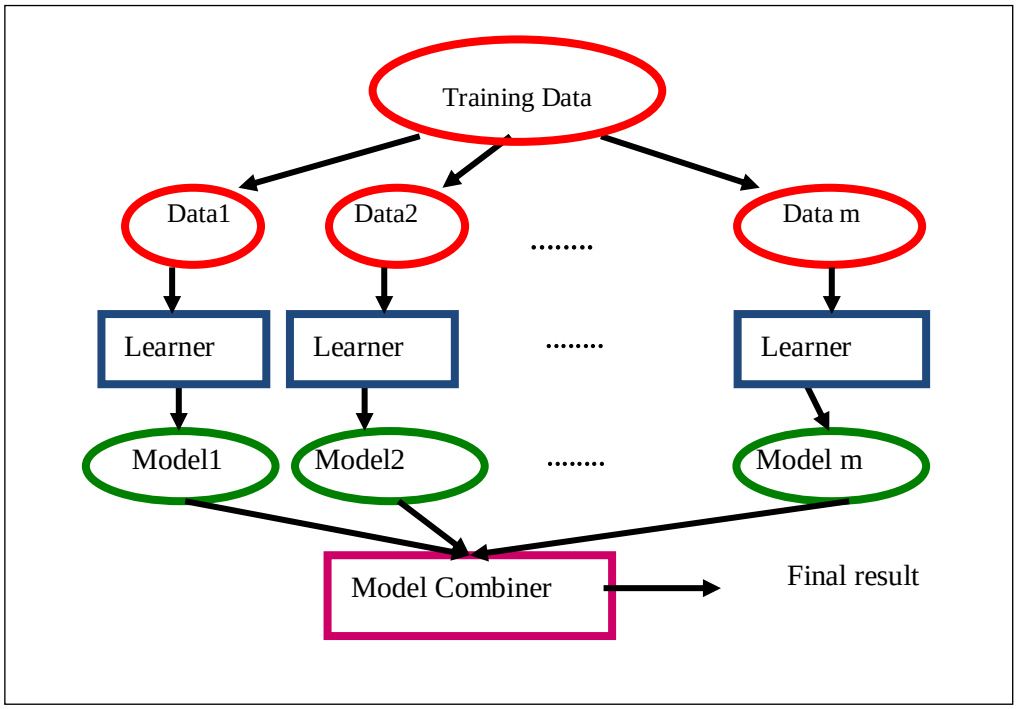


Figure 3.4: *Costing* algorithm based on Cost-proportionate rejection sampling with aggregation.

Several literature reviews show different sampling methods, where some of them amend the number of negative examples (over-sampling); some of them change the number of positive examples (under-sampling); a few of them use the *SMOTE* (Synthetic Minority Over-Sampling Technique) algorithm that tackles the imbalanced problem by generating synthetic minority class examples (Chawla, 2002). Kubat and Matwin (1997) used one side selection by under-sampling the majority class, while keeping the original population of the minority class. In addition, *CSRoulette* (Sheng and Ling, 2007) is similar to *Costing*, except that Costing uses cost proportional rejection sampling, while *CSRoulette is* based on the cost proportional roulette sampling.

Drummond and Holte (2003) demonstrated that under-sampling outperforms over-sampling for imbalanced class distribution and unknown cost ratio, and their experiments show that

this is because the over-sampling has little sensitivity to changes in the misclassification cost than under-sampling. Furthermore, Maloof (2003) compared cost-sensitive learning methods to sampling, but found that cost-sensitive learning, over-sampling and under-sampling performed nearly identically.

3.2.1.2.2 Thresholding

Thresholding is a very simple cost-sensitive learning method and is applicable to any classifiers such as decision tree, neural network, and Naïve Bayes. It can convert a cost insensitive learning classifier to a cost-sensitive learning classifier. Thresholding is the process for searching for the best threshold and predicting the testing set according to the optimal threshold. In fact, this method is based on a threshold to classify examples into positive or negative if the cost-insensitive classifiers can produce probability estimations. It works by selecting a threshold, which is probability estimated on training instances that minimizes the misclassification cost, then, uses that threshold for predicting testing instances (Sheng and Ling, 2006). Sheng and Ling (2006) divided thresholding methods into two categories, *theoretical thresholding*, and *adjusted thresholding* as the following:

- ***Theoretical thresholding***

  In particular, Elkan (2001) used the *theoretical threshold* to determine the optimal decision for reducing the expected cost. This method can be achieved by multiply the number of negative (Frequent) examples in the training to rebalancing the training dataset. A *target probability threshold* P' is defined and would be achieved correspond to a given probability P, where P is a *theoretical threshold* for making an optimal decision on classifying instances into rare examples. Therefore, the number of frequent examples should be multiplied by equation (3.12):

  $$\frac{p'}{1-p'}\ \frac{1-P}{P} \qquad (3.12)$$

  More precisely, to rebalance datasets, it is typically P= 0.5. Thus, the number of frequent examples will be multiplied with just $\frac{P\prime}{1-P\prime}$ where it is equal to $\frac{FN}{FP}$. Consequently, Elkan (2001) used this theorem to reduce a cost, by multiplying $\frac{FN\ (false\ frequent)}{FP\ (false\ rare)}$ with the number of negative examples (frequent class). Mathematically, his theorem changes the number of frequent examples without

duplicating or discarding any of the rare examples. This type of threshold makes the optimal decision for classifying instances into positive class (rare class).

- ***Adjusted thresholding***

Sheng and Ling (2006) suggest this type of threshold, where this thresholding searches for the best probability estimated on the training instances, then uses it for future predictions of testing instances; and if the test instance with predicted probability above or equal to this threshold is classified as positive (rare class); otherwise as negative (frequent class). The function of the threshold represents the misclassification cost function as given in equation (3.13).

$$Misclassifiaction_{cost} = function(Threshold) \quad (3.13)$$

More precisely, to choose the best threshold, one only needs to calculate all the misclassification costs for each possible probability estimates on the training examples, then we will get the curve of the thresholds as shown in Figure 3.5, finally choosing the best threshold that minimizes the total misclassification cost which is the valley point in the curve.

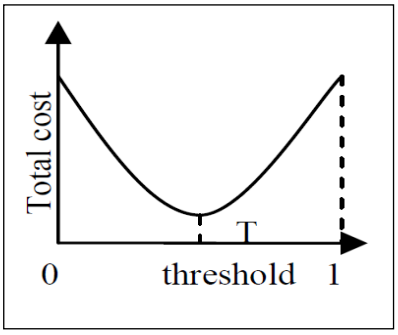


Figure 3.5: The best threshold is the point that gives minimum cost (Sheng and Ling, 2006).

Experimentally, Sheng and Ling (2006) show that adjusted threshold is highly effective. On the other hand, theoretical and adjusted thresholding is best when the cost ratio is large. As a result, the only problem in the adjusted threshold is that it is time consuming to search for the best threshold.

3.2.1.2.3 Weighting

This category is based on assigning a weight, which is based on a misclassification cost, to each example to reflect its importance. For example, if the cost of the misclassification for

class i is 4, and the cost of the misclassification for class j is 1, then a weight of 4 is assigned to examples of class i and a weight of 1 to the examples of class j is 1. Thus, more weight is given to those examples with the higher misclassification cost. Hence, an error-based learner that uses weights can use this information to concentrate on important examples.

One such algorithm is C4.5CS (Ting, 1998; 2002), which is similar to the *GINI* Altered priors method (described above in Section 3.2.1.1), except that GINI Altered priors does not use the weights when pruning. Where pruning is a process of removing nodes or sub-trees aimed at reducing the effect of statistical noise or variation that may be based on a training set. Other algorithms that use this idea include *MaxCost* and *AvgCost* (Margineantu and Dietterich, 2003). However, both *MaxCost* and *AvgCost* have been designed to solve multi-class problems, where *MaxCost* uses the worst or maximum cost of misclassifying an example of a given class; which is the maximum value within the column representing the actual class in a cost matrix. While *AvgCost* calculates the average cost of misclassifying an example, which can be obtained by computing the average cost of the values in the column representing the actual class value. The following equations summarise theses three weighting methods:

$$\text{C4.5CS:} \qquad \text{weight}_j = \text{Cost}(j)\frac{N_j}{\sum_{i=1}^{k}\text{Cost}(i,j)*N_i} \qquad (3.14)$$

$$\text{MaxCost}: \qquad \text{weight}_j = \underset{1\le i\le k}{\text{Max}}\ \text{Cost}(i,j)$$

$$\text{AvgCost}: \qquad \text{weight}_j = \frac{\sum_{i=1,i\neq j}^{k}\text{Cost}(i,j)}{(k-1)}$$

Where, Cost(j) is the misclassification cost of class j, and k is the number of classes, in the multi-class cost matrix when i is predicting the column and j is the correcting column (actual class).

3.2.1.2.4 Relabeling

Relabeling involves considering whether the class of training or the testing instance should be changed to reflect the costs of misclassification (Michie et al., 1994). The relabeling method can be divided into two categories: *relabeling the training instances* and *relabeling the test instances*:

- Relabeling the training instances: such as *MetaCost* (Domingos, 1999), which will be described in section 2.3.1.2.5.

- Relabeling the testing instances: such as in *CostSensitiveClassifier*(CSC) (Witten and Frank, 2005), by predicting the class with a minimum expected misclassification cost, rather than the most occurred class. Performance can often be improved by using a Bagged classifier to improve the probability estimates of the base classifier, that will be described in section 2.3.1.2.5.

3.2.1.2.5 Ensemble learning methods

Ensemble learning combines multiple independent models with the aim of producing better classifiers. The ensemble learning approach depends on learning from a single model *base learner* and then the predictions of those base learners are combined by using voting, weighting or averaging. In the data mining WEKA software, the ensembles method is called *meta-learners*, which is based on taking a learning algorithm as the base learner, and creating a new learning algorithm. Practically, there are two approaches to ensemble learning: *bagging*, and *boosting*, which are described below:

i. Bagging

Bagging, introduced by Breiman (1996), involves three steps:

- Creating *m* ensembles (booststrap samples); by drawing *n* examples randomly re-sampling the training data with replacement from the original data.
- Applying a specific learning algorithm (base learner) independently to the different samples to generate different models.
- The different models are aggregated by using the average in the case of regression, and voting in the case of classification; by combining the *m* resulting models using a simple majority vote, to predict an unseen instance.

  Thus, bagging also called ***B****ootstrap* ***agg****regating*. Examples of cost-sensitive bagging algorithms include *MetaCost* (Domingos, 1999), which uses relabeling, *CostSensitiveClassifier* (Witten and Frank, 2005), *Costing* (Zadrozny, 2003a; 2003b) which was described above in Section 3.2.1.2.1. In particular, the idea with *MetaCost* is summarized by the Figure 3.6, *MetaCost* has the following four steps:

  1. Generate *n* samples with replacement from the training data.
  2. Apply the base learner on each sample to produce *n* classifiers.

3. Estimate the expected cost of misclassification, and relabel each example of the training data with a new class label that minimizes the expected cost.

4. Finally, use the base leaner on the relabelled training data to generate a cost-sensitive classifier.

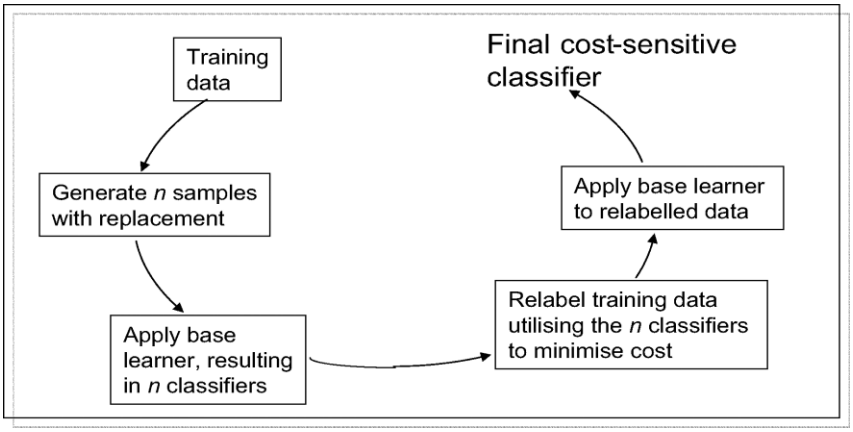


Figure 3.6: The *MetaCost* system (Domingos, 1999)[1]

Also, the other type of bagging is *CostSensitiveClassifier* (Witten and Frank, 2005), which can belong to weighting or relabeling methods as well. *CostSensitiveClassifier* is a meta classifier that makes its base classifier cost-sensitive, two methods can be used to introduce the cost:

- *Weighting*: By reweighting training instances according to the total cost assigned to each class.
- *Relabeling*: By relabeling the test instances; predicting the class with minimum expected misclassification cost, rather than the most occurred class. Performance can often be improved by using a Bagged classifier to improve the probability estimates of the base classifier.

ii. Boosting

Boosting was introduced by Schapire (1999) and in response to a question posed by Kearns (1988) "*Can a set of weak learners create a single strong learner?"*. The process of boosting is carried out in a sequential manner in different turns, and at the end of each

[1]Figure taken from (Vadera, 2010).

turn, the weights are adjusted to reflect the importance of the instances for the next learning turn. The boosting approach is based on reweight training data. It involves creating a number of hypotheses h_t and then combining them to form a more accurate composite hypothesis. The following four steps summarise the boosting process:

- Weight each example in the training dataset; by giving a higher weight to examples that have higher misclassification costs, and lower weights to examples that have a low cost.
- Applying a classifier with the new weights.
- Checking the example on the classifier, to see whether the predicted class matches the actual class and change the weight of the examples by increasing the weights of misclassified examples, after that the new weights are passed to the next of boosting.
- After many iterations and using the first three steps; combine the different hypotheses and determine the final prediction class; which is a strong learner that is well correlated with the true classification.

One of the earliest examples of the use of boosting is *AdaBoost* (Adaptive Boosting) (Freund and Schapire, 1996) which used an accuracy based learner to generate an improving sequence of hypotheses. *AdaBoost* starts the boosting process by assigning unit weights to each example. Then in each sequential trial, increases the weight of misclassified examples and decreases the weight of the other examples; it assigns the same weight in the first turn which is 1/N, where N is the total number of instances, then the weight changes over different classification turns according to misclassification errors. After many sequential trials, it combines these hypotheses to perform final the classification, which is based on selecting the class that results in the maximum weighted vote as illustrated in Figure 3.7. Schapire (1999) introduced equation (3.15) to form a more accurate composite hypothesis, as shown in Figure 3.8:

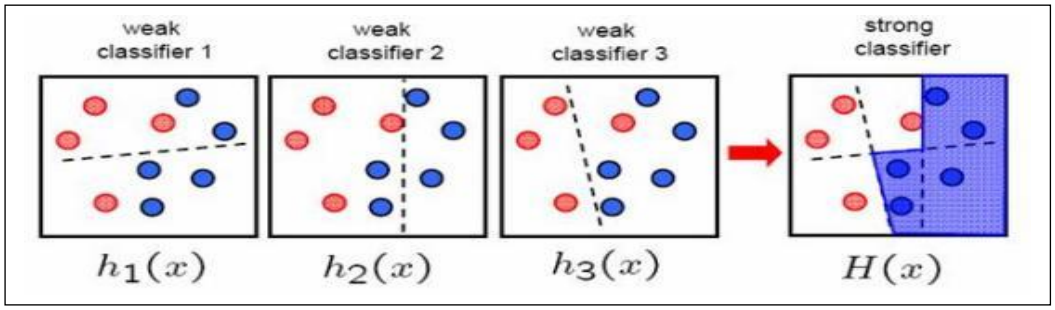


Figure 3.7: Illustration of boosting method (UCSD, 2015).

$$f(x) = \sum_{t=1}^{T} \propto_t \quad h_t\,(x) \qquad (3.15)$$

Where $\propto_t$ represents the extent of the weight given to $h_t(x)$ in each time *t*.

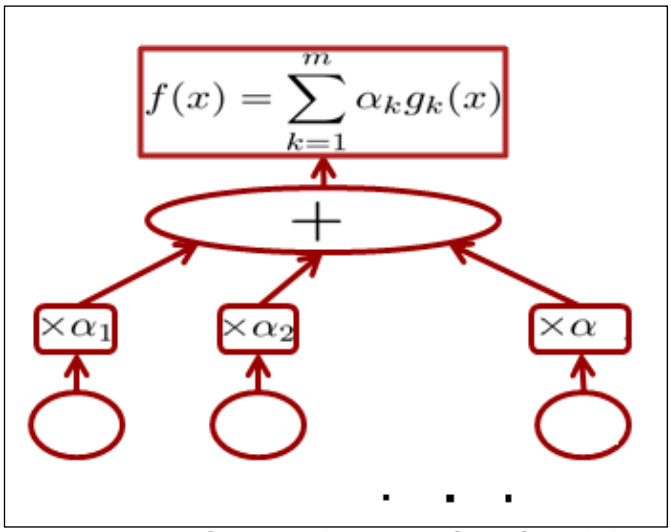


Figure 3.8: Cost-sensitive boosting (composite hypothesis) (UCSD, 2015).

There are several studies that use boosting and modify the weight rules to take account of costs, including *AdaCost* (Fan et al., 1999), *Cost-UBoost (*Ting and Zheng, 1998a), and *GBSE* (Abe et al., 2004). Whereas, *AdaCost* uses the cost of misclassifications to update the training distribution by assigning high initial weights to costly examples, then increases the weights of costly misclassifications more but decreases the weights of correct classification less.

3.2.1.3 Algorithms that use optimization methods

Genetic algorithms (GAs) have been utilized by several authors to learn cost-sensitive decision trees. One of the first studies was by Turney (1995), who developed *ICET* (In*expensive Classification with Expensive Test*) which uses GAs to evolve decision trees in order to minimize both test costs and misclassification costs. ICET uses a genetic pool that consists of genes representing the cost of attributes (CA), biases ***ω*** (parameter used to control the amount of weight which should be given to the cost), and parameters CF (parameters used to indicate the level of pruning by C4.5). These parameters are used in a version of C4.5 to generate trees, where the information gain measure as shown in equation 3.4 is replaced with an *Information Cost Function* (ICF_A) for an attribute A (that modifies the information gain formula to include costs and is adopted from EG2 (Nunez 1991) as shown in equation (3.5).

Thus in ICET, trees are not represented in a genetic pool directly, but are actually learnt using the genes as parameters for a decision tree learner that uses EG2's *cost function*, instead of an information gain, to generate a decision tree for each individual, as shown in Figure 3.9. Following this process, all of these decision trees are evaluated using expected costs as a fitness function, and a new pool is produced using mutation and cross over. This process is repeated 50 times and the fittest tree returned.

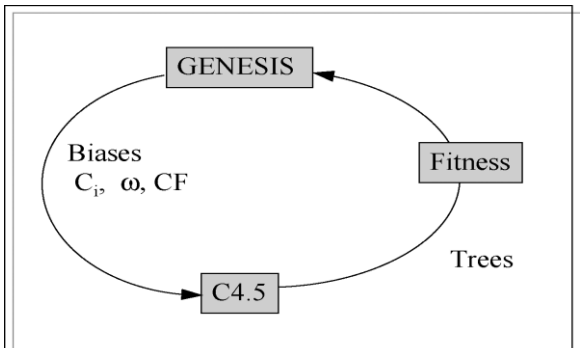


Figure 3.9: The ICET System (Turney1995)[2].

In contrast, Omielan and Vadera (2012) developed *ECCO (Evolutionary Classifier with Cost Optimization)* that functions directly through a pool of decision trees that is represented by the genes as bits of string, which are used to construct the decision trees. Their comparison with ICET suggests that ECCO is more cost-sensitive and effective than ICET.

3.3 Literature review of research on cost-sensitive Bayesian network algorithms

Historically, most of the cost-sensitive algorithms developed have focussed on learning decision trees, with a recent survey comparing over 50 algorithms (Lomax and Vadera, 2013). In contrast, little attention has been paid to developing cost-sensitive Bayesian networks, which are (Gao et al., 2008; Nashnush and Vadera 2014; Jiang et al., 2014; Kong et al., 2014). This section presents a literature review of research aimed at developing algorithms that learn cost sensitive Bayesian networks.

The first attempt was in (2008), when Gao, Wang, and Cheng introduced a cost sensitive loss function for estimating parameters. As described in Chapter 2 in Section 2.4.1 (Bayesian network structure learning), one approach to learning Bayesian networks is to perform a

[2] Figure taken from (Vadera, 2010).

search that optimizes a score function such as *MDL* (Rissanen, 1978), *AIC* (Akaike, 1974), *BIC* (Schwarz, 1978), and these functions do not include costs. Thus, Gao et al., (2008) suggested amending the cost-insensitive objective function to include costs. The cost-insensitive Log-Likelihood loss function, described in Section 2.4.1.1, takes the form from extended probability, which represents as an equation (3.16) instead of normal probability; in particular, to include the cost, the value $P(X_i|X_j)$ is extended to:

$$P(X_i|X_j)^{-Cost(3-k,\ k)} \qquad (3.16)$$

Where, $X_i, and\ X_j$ are two nodes in a BN, and k represents the number of a class label, where they focused on two class problems; when k=2. Thus, Gao et al. (2008) amended the function in equation (2.8) to the following *Cost Sensitive Loss function* (CSL):

$$\mathrm{CSL}(X_i|X_j) = -\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{k} Cost(3-k,\ k) * \log P(X_i|X_j) \qquad (3.17)$$

Where, $Cost(3-k,\ k)$ is the cost of misclassifying an instance. The new cost-sensitive Bayesian networks algorithms are applied during the learning structure, but they do not amend the probably during the learning of the parameters. In addition, their new cost-sensitive Bayesian networks algorithms are evaluated by comparing their algorithms with existing cost-insensetive algorithms. Experimentally, they carry out an empirical evaluation of this method on a two class problem, and it shows that cost-sensitive Bayesian networks with cost sensitive loss function are effective compared with the cost in-sensitive Bayesian networks.

However, they do not evaluate their work with existing cost-sensitive algorithms like *MetaCost* or other cost-sensitive classifier, so the claim is not fully substantiated.

More recently, Jiang et al. (2014) used an instance weighting method inspired by the approach used by Ting (2002). Where, they modify the probability estimate that is used in learning parameters (that described in Section 2.4.2 in equation (2.17)) by incorporating the instance weights (that described in Section 3.2.1.2.3 in equation (3.14)). The weights they adopt are presented in equation 3.18:

$$P_w(X \mid C_j) = \frac{W_j * \ (N_j+1)}{\sum_{i=1}^{k} W_i * N_i \ + n_X} \qquad (3.18)$$

Where, W_j is the weight of class j instances; k is the number of class labels; n_X is the number of values of node X. These weights, which include costs, are utilized during estimating the probabilities (i.e., parameters). Their results shows that the performance of cost-sensitive Bayesian networks is good when the cost ratio is large. However, they change the probability at the last stage during learning parameters after learning the structure so do not take account of costs when learning the structure. Also, their experiments have been compared with the original BN classifier, but not with other cost-sensitive classifiers such as *MetaCost* classifier (Domingos, 1999). In addition, their experiments are based on just four cost ratios which are {2,5,10,and 15}.

The most recent research in this field uncovered in the literature search is by Kong et al. (2014). They developed a cost-sensitive Bayesian network classifier, and then applied it on real-world *rock burst prediction examples*. Their algorithm is based on the concept of adjusting thresholds (described in Section 3.2.1.2.2) due to Sheng and Ling (2006). This algorithm starts by learning a cost-insensitive Bayesian network structure from a training dataset. Then, each instance in the testing set is classified to the class label that minimizes the expected cost. This cost-sensitive classifier provides a simple effective method for rock burst prediction. Their approach is compared with the usual cost-insensitive Bayesian network classifier but they do not compare it with other cost-sensitive methods.

Table 3.3 summarises the literature review and contrasts the different methods for learning cost-sensitive Bayesian network algorithms discussed in this chapter.

Authors	Approach	Aims	Weakness
(Gao et al.,2008).	Direct methods by amending the LL loss function.	Include misclassification costs during learning structure, it aims to minimize the misclassification costs.	• They do not use the same cost to learn parameters of a structure. • They do not evaluate their work with existing cost-sensitive algorithms like *MetaCost* or other cost-sensitive classifier. • They used constant cost matrix 1:5
(Jiang et al., 2014)	Modify the *simple estimate* of a probability by using a weighting method to weight instances	Reweight instances according to the misclassification cost.	• They do not include any costs when learning structure. • They do not compare their experiments with other cost-sensitive classifiers. • Their experiments are based on just four cost ratios which are:{2,5,10,and 15}.
Kong et al., 2014).	*Adjusted thresholding*	Aims to minimizing the misclassification cost.	• Their approach is compared with the usual cost-insensitive Bayesian network classifier but they do not compare it with other cost-sensitive methods • Their experiments are on a specific application (*rock burst prediction examples*).

Table 3.3: Summary of the literature review of cost-sensitive Bayesian network algorithms.

3.4 Summary

This chapter has presented a comprehensive survey of existing cost-sensitive learning algorithms. The chapter started by defining the difference between cost-insensitive algorithms, and cost-sensitive algorithms.

The survey was in two parts. The first part is, the field of cost-sensitive decision tree learning, was surveyed by many different algorithms and approaches. In particular, it revealed that cost-sensitive decision tree algorithms are based on three methods; direct methods, indirect methods, and the use of optimization methods.

The second part of the survey focused surveying cost-sensitive Bayesian networks. This showed that there are just three studies aimed at addressing this problem, all of which are very recent when compared to the studies on decision trees. These methods were contrasted and summarised in Table 3.3.

In the next chapter, new cost-sensitive Bayesian network learning algorithms will be proposed based on the types of approaches uncovered when developing cost-sensitive decision trees.

Chapter 4: Cost-Sensitive Bayesian Network Learning Algorithms

In the previous chapter, it was noted that three different strategies have been used for developing cost-sensitive decision tree learning algorithms: (i) direct method, (ii) indirect method, and (iii) optimization methods. Hence, in this chapter we describe how these strategies are used to develop cost-sensitive Bayesians network algorithms. Section 4.1 presents the use of an indirect method to develop a new algorithm for learning cost-sensitive Bayesian networks using sampling approach. Section 4.2 develops a new algorithm for learning cost-sensitive Bayesian networks by using a direct method to amend an existing cost-insensitive algorithm to include costs directly into algorithm's process. Section 4.3 presents the development of a new cost-sensitive Bayesian networks algorithm based on using Genetic algorithms. Section 4.4 presents a discussion and summary about the proposed algorithms in this chapter.

4.1 Learning cost-sensitive Bayesian networks via a sampling approach

This section presents the use of an indirect method to develop a new cost-sensitive Bayesian networks learning algorithm by using a *sampling approach* to take account of misclassification costs. As described in Section 3.2.1.2, indirect methods do not change the learning process of a classifier but use the classifier as a black box.

The approach used is based on a *Folk theorem* that introduced by Zadrozny et al. (2003a) and Elkan (2001) that was described in Chapter 3, in Section 3.2.1.2.1. This theorem draws a new distribution from the old distribution, according to misclassification costs to change the data distribution and obtain an optimal cost-minimization from the original distribution. In particular, the data distribution can be changed to reflect the costs (see equation (3.10), and (3.11) in Chapter 3 for a description).

The *Folk theorem* can be used to create a new distribution from the old distribution by multiplying the old distribution with a factor proportional to the relative cost of each example. For example, consider the hepatitis dataset, which has 32 instances in the class "Die" (class distribution 20%), and 123 instances in the class "Live" (class distribution 80%). Given the imbalance in examples for the two classes, an accuracy based classifier will always

be biased to the most common class, that is "Live", though misclassifying examples of class "die" is more serious. The *Folk theorem* can be used to address this kind of situation. Suppose the misclassification costs are 4:1 for "Die": "Live" respectively. Then, using equation (3.4) to change the data distribution, thus, the new distribution of class "Die"=4*32 = 128 instances (class distribution 50%); while the new distribution of class "Live" =1*123 = 123 instances (class distribution 50%). Figure 4.1 summarised the steps of using *sampling approach* with hepatitis dataset example.

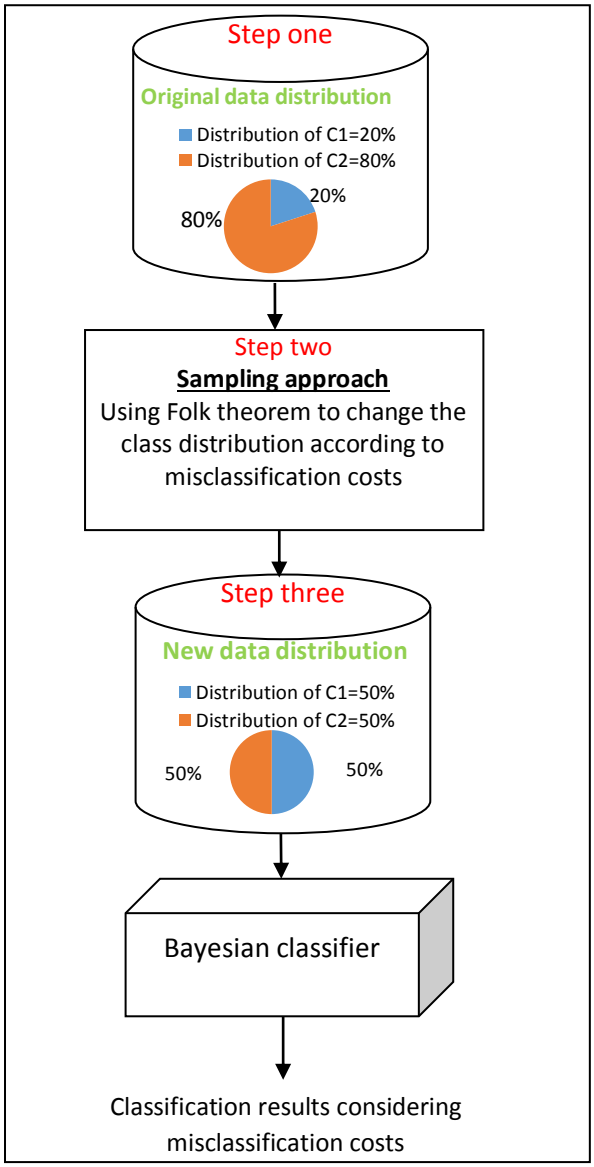


Figure 4. 1: Illustration of sampling approach steps with hepatitis dataset.

Folk theorem draws a new distribution from the old distribution, according to cost proportions to change the data distribution and obtain optimal cost-minimization from the

original distribution. Figure 4.2 presents our new algorithm that called *Cost-Sensitive Bayesian Network* (*CS-BN*) algorithm via sampling approach.

CS-BN via sampling approach (indirect method)

1. Divide dataset into 75% of instances for training, and 25% for testing.
2. Change the training data distribution according to the misclassification cost in each class:

$$\frac{N_1'}{N_2'} = \frac{N_1 * C_1}{N_2 * C_2}$$

3. Learn the TAN structure and its parameters.
4. Evaluate the TAN on the original test set distribution.

Figure 4.2: CS-BN algorithm using sampling.

The main steps of this algorithm are:

Step 1: The data are split into a training set and testing set. The training set uses 75% of the original data, while the testing set uses 25% of the original data[3].

Step 2: The distribution of the data is altered to take account of costs. The *Folk theorem* is used to change the data distributions (as described in Chapter 3 in equation 3.11). For example, as we described above, in the hepatitis dataset, where the number of examples that belong to class "Die", and "Live" are 32, and 132 respectively, the old distribution is 20%: 80% respectively. If the relative costs that used in Chapter 3, Table 3.1 are £50:£1, the *Folk theorem* can be used to change the distribution as follows:

For class "Die"= (50*32)/((50*32)+(1*132)= 1,600/1,732=0.92

For class "Live"= (1*132)/(50*32)+(1*132)= 132/1,732 = 0.08

Thus, the number of class "Die" to class "Live" will be changed to 1,600 to 132 respectively, and the distribution will be 92% to 8% respectively.

[3] Other ways of splitting the data could, of course be adopted without affecting the principles of the approach.

There are different methods that can be used to sample the data and redistribute the data. During our research, we used two methods, under-sampling and over-sampling as described in Section 3.2.1.2.1. When the new proportion is less than the original proportion, under-sampling (without replacement) is used to delete some of the examples in the frequent class. On the other hand, if the new class proportion is greater than the original class proportion, over-sampling (with replacement) is used to randomly select new instances which belong to the rare class, and hence increase the number of examples. In particular, Elkan (2001), and Zedrzony (2003a) mentioned that, this method is the most affected method that used to reflect misclassification costs; where using under-sampling to delete some of unimportant examples , and using oversampling to duplicate some of important examples.

Step 3: Once the data is redistributed, Friedman et al.'s (1997) algorithm is used to learn Tree Augmented Bayesian Networks, (as given in Section 2.4.1.3.2).

Step 4: The learned TAN is evaluated using the testing data from the original distribution. The measures used are the accuracy and expected cost (as given in Section 3.2 in equations (3.1) and (3.2)).

4.2 Learning cost-sensitive Bayesian networks via an amending approach

This section presents a direct method to developing cost-sensitive Bayesian network algorithm by amending an existing cost-insensitive algorithm.

The approach adopted is motivated by the use of direct methods for developing cost sensitive decision tree learning algorithms, which described in Chapter 3, Section 3.2.1.1. In particular, a key step in decision tree learning is to select the criteria used for the next node of the decision tree. Early decision tree induction algorithms that focused on accuracy used a measure based on information theory to select the splitting criteria. For example, *ID3* and *C4.5* (Quinlan, 1979) are based on calculating the gain in information achieved by each of the attributes if these were chosen for the split and choosing the attribute which maximizes this gain. Thus, an obvious way of adapting these algorithms is to adapt this measure to take account of costs. For example, Breiman et al.(1984) modify the class probabilities P(i), that are used in the information gain measure, and replace that probability with the altered probability as shown in equation (4.1) where the probability P(i) is weighted by the relative **Cost ratio$_i$** as follows:

$$\text{Altered Probability i} = \text{Cost ratio}_i * \left(\frac{Ni}{N}\right) \text{ , where Cost ratio}_i = \frac{\text{cost (i, j)}}{\sum_j^k \text{cost(i, j)}} \quad (4.1)$$

Where, Ni is the number of examples in class i, while N is the total number of examples. Cost ratio_i represents the ratio of the cost of class i to the total costs, k is the number of classes, where this equation are applicable in just two class problems. For example, for the cost matrix in Table 3.1, the cost ratio for the positive class is 50/51, while, the cost ratio for the negative class is 1/51.

Pazzani et al. (1994), also use this approach but for a different splitting criteria known as the GINI index. Figure 4.3 illustrates this idea when there are two classes C1, and C2, and each class given particular attribute values AttV1 and AttV2. Initially, two classes have an equal chance of occurring (i.e. probability of 0.5) and are altered to have probabilities of 0.75 and 0.25 respectively, to reflect a misclassification cost ratio of 3:1.

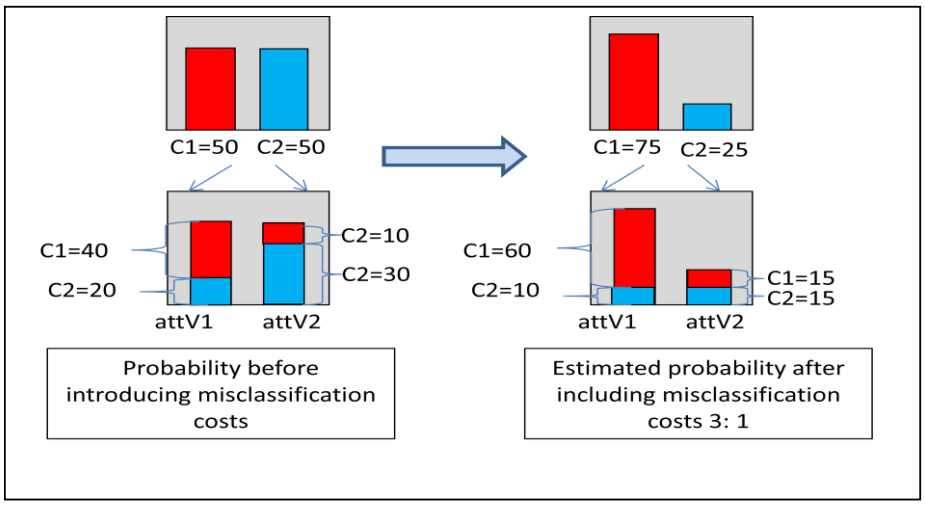


Figure 4.3: An illustration of the altered probability.

Our algorithm called *Cost-Sensitive Bayesian Network* (*CS-BN*) algorithm via amending approach is based on the following question, *"how can a similar approach be used for amending an existing Bayesian network learning algorithm?"* . As mention in Section 2.4.1.1 learning a Bayesian network structure requires searching for the best network according to a score function. Many scoring criteria have been described, including the *minimum description length* (MDL) which is defined by equation (2.11) (see Chapter 2, Section 2.4.1.1 for more details). As described in Section 2.4.1.3.2, a key step of existing algorithms is to compute the Minimum Description Length (MDL) while learning the Bayesian network structure. Hence, by analogy with the approach take for decision trees, where the information

theoretic measure was modified, the modification made to develop our new algorithm is to change the original MDL measure. We make two amendments: (i) when learning the structure of a Bayesian network, in MDL equation because this equation can determined the strongest links between nodes, also it is the key step of Bayesian networks algorithm, and (ii) when learning the parameters of the structure, in simple estimator equation because this equation can determined the relationships between nodes, also, by analog with what others have done in decision tree, where some researches are based on amending the probability estimation to include the costs .These amendments are described in Section 4.2.1 and 4.2.2 respectively.

4.2.1 Amending the formula for learning the structure

First, the *Log-likelihood* factor that is used in the *MDL* measure in Chapter 2, in equation (2.8), is amended to take account of costs. The modification made is to multiply each part of the information measurement with the cost ratio of a class, and the new Log Likelihood function $\mathrm{LL}(X_i|X_j)$ is as shown in equation (4.2).

$$LL(X_i|X_j) = \sum_{i=1}^{n}\sum_{\mathrm{j}=1}^{m}\sum_{\mathrm{k}=1}^{2} \mathrm{p}(X_i, X_j) \quad \log\left(\frac{\mathrm{p}(X_i, X_j)}{\mathrm{p}(X_j)}\right) * \text{ Cost ratio }_{\mathrm{k}} \qquad (4.2)$$

Where, K is the number of class labels, *n*, and *m* represent the order of connected nodes. While $\mathrm{p}(X_i, X_j)$ represents the probabilities of events X_i, X_j happened in D. While, $\text{Cost ratio}_{\mathrm{k}}$ is the ratio of misclassifying class k over the total costs, as described in equation (4.3).

$$\text{Cost ratio}_{\mathrm{k}} = \frac{\text{cost k}}{\text{Total costs}} \qquad (4.3)$$

4.2.2 Amending the formula for learning parameters

Secondly, the parameter estimator that described in Chapter 2, Section 2.4.2 in equation (2.17) is modified to reflect misclassification costs by modifying the conditional probability of each node given its parent. That is, instead of using the simple estimator of probability we weight it by the cost ratio:

$$P(x_i \mid \pi_{xi}, C_k) = \text{Cost ratio}_k * \frac{p(x_i, \pi_{xi}, C_k) + \alpha}{p(\pi_{xi}) + (\alpha * n_{X_i})} \quad (4.4)$$

Where, x_i is the node that is connected with its parents (class label C_k, and another parent π_{xi}); n_{xi} is the number of the possible values of node xi. While, $\alpha = 0.5$ represents the initial count on each value to avoid 0.

These amendments lead to the algorithm presented in Figure 4.4, and described in detail below; Where the first :

CS-BN via amending approach (direct method)

1. Compute new conditional LL information between each pair of attributes (nodes) based on class label, and include cost ratios for each class in the calculation:

$$\sum_{x}^{n}\sum_{y}^{m}\sum_{k}^{2} p(x, y, \text{Class}_k) \log\frac{p(x, y, \text{Class}_k)}{p(y, \text{Class}_k)} * \quad \text{Cost ratio}_k$$

2. Build a complete undirected graph between each pair of attributes (nodes) without class node.
3. Using the Maximum Weight Spanning Tree algorithm, to maximize the information gained about the classification weighted by the cost of misclassification obtain a tree.
4. Convert the tree to a directed tree.
5. Add the class label as root for all attributes (nodes).
6. Learn the parameters for each node with its parents by using the new probability estimation that includes misclassification costs.

$$p_c(X| Y, C_K) = \text{Cost ratio}_k * \frac{P(x, y, C_k) + \alpha}{P(y, C_k) + (\alpha * n_X)}$$

Figure 4.4: CS-BN algorithm using the amending approach.

This new algorithm was implemented in *Java NetBeans* using the data mining software WEKA open source to help in the development and implementation. Also, an empirical

comparison with existing algorithms (standard Bayesian networks approaches; *MetaCost+BN*, and *MetaCost+J48*. This is presented in Chapter 5: an empirical evaluation of the new algorithms for learning cost-sensitive Bayesian networks. In our experiment, we use the same original statistical formula (Friedman et al., 1997), but we change the formula to include the cost ratio of each class, by multiplying each part of information measure with cost ratio of class. The main steps of this algorithm are:

Step 1: Compute Conditional Information

The first step is calculates the information between each node and all other nodes, by using MDL score that based on the new Likelihood function LL(B|D) that is given in equation (4.2).

Step2: Build a complete undirected graph

An undirected graph is constructed, where the nodes are the attributes of data and the edges represents the information (dependencies) between nodes. The weights on the edges represent the extent of the dependencies, adjusted by the relative cost as calculated in Step 1.

Step3: Apply (MWST) algorithm

Find a maximal weight spanning tree between nodes by running a maximum-weight spanning tree (MWST) algorithm (Cormen et al., 1990) to obtain undirected graph.

Step4: Convert to directed tree

The undirected graph is converted to a directed graph by choosing the root of the first maximum connection in the previous step, then adding a direction to the next connection if it does not lead to a cycle. This process is repeated until all the nodes have been considered.

Step5: Add the class label as root

The class label node is added as the parent (root) node for all nodes.

With these 5 steps the Tree Augmented Naive Bayes structure will be created.

Step6: Learn the parameters

After creating the structure of a TAN, the last step is to learn the parameters for each node with its parents by using equation (4.4).

4.3 Learning cost-sensitive Bayesian networks via Genetic algorithms

This section develops an algorithm for learning cost-sensitive Bayesian networks that utilizes Genetic algorithm, where the *genes* are utilized to represent the links between the nodes in Bayesian networks, and the expected cost is used as a *fitness function*.

4.3.1 Encoding tree augmented networks

The structure of a TAN can be viewed as a directed graph which can be represented as an adjacency matrix A. Where an element A(i,j) is set to "1" if node j is a parent of node i, and set to "0" if there is no links between node j and node i. Figure 4.5 illustrates the idea, where node *a0* has two parents which are *a2* and *a4*, and hence A(0,2)=A(0,4)=1; while it does not have any links with a1and *a3* so A(0,1)=A(0,3)=0.

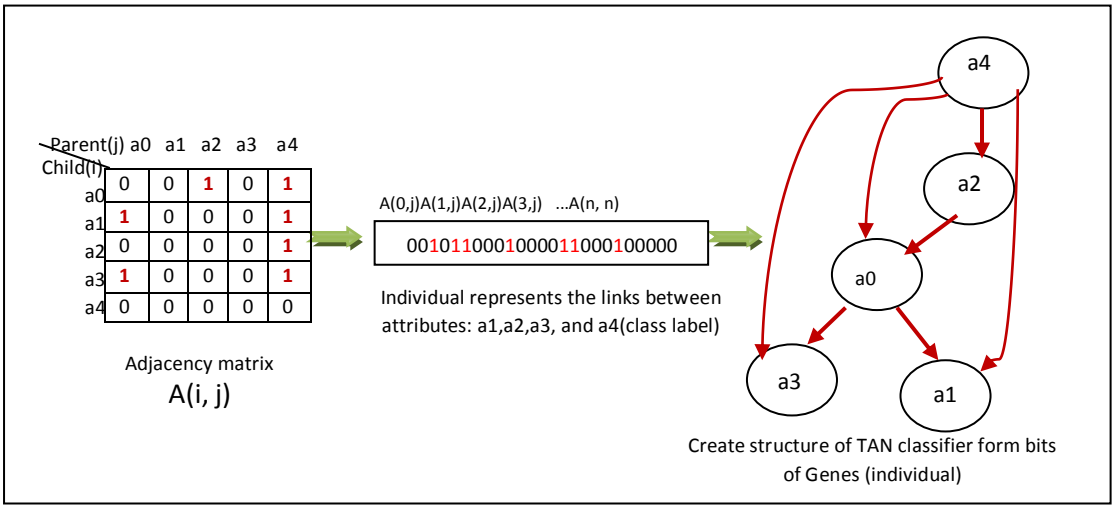


Figure 4.5: An illustration of how TAN classifier represents the genes.

To generate the initial pool of TAN trees for a GA involves three steps: firstly, generating the adjacency matrix randomly; secondly, testing the adjacency matrix to ensure that it denotes a valid TAN, and if not to make it a TAN; and thirdly, converting the adjacency matrix to a linear string of bits that can be used by a GA. These three steps are described and illustrated below:

- Firstly, when generating the initial population or following mutation or cross over, it is possible to obtain an illegal TAN as shown in Figure 4.6 (a).

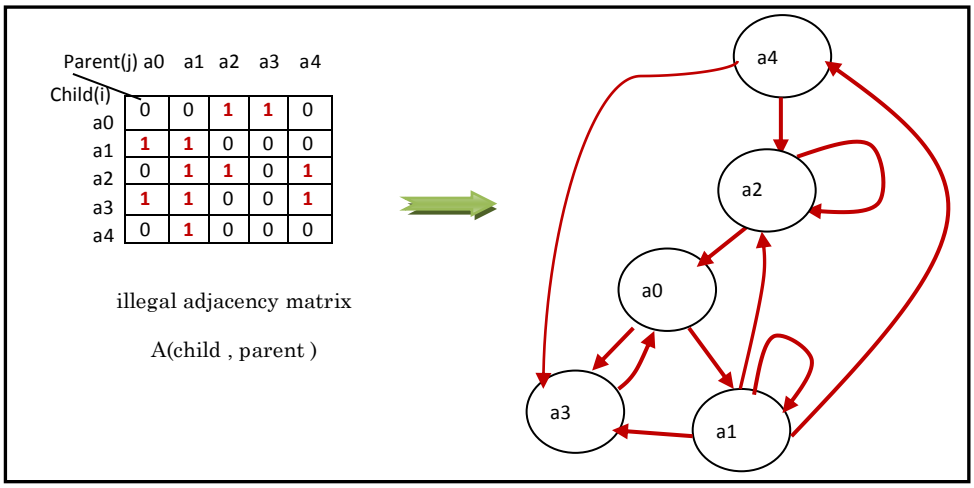


Figure 4.6 (a): An illegal TAN structure, created from adjacency matrix A(child, parent) in CS-BN via GAs

- Secondly, to make sure we have legal TANs, we check the following three conditions:

i. There must be no immediate circularity on each node, where a node i is a parent of itself. If this is the case, then A(i,i) is set to zero as illustrated figure 4.6 (b) :

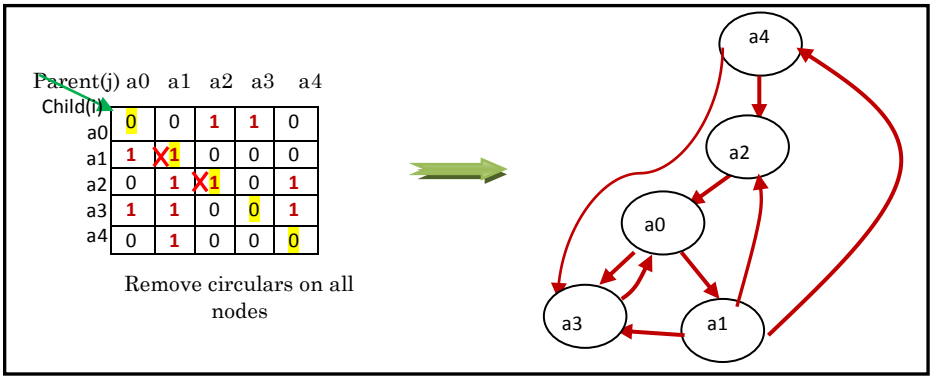


Figure 4.6(b): There is no circular on each node A(i, i) =0 in CS-BN via GAs

ii. By definition, for a TAN; the class node must have no parents, and all the other nodes must have the class node as a parent and one other parent that is chosen from the other nodes. If this is not the case, then this is corrected by making sure the class node is added as a parent, and one of the other nodes is chosen randomly as the other parent, where, Figure 4.6 (c) illustrates the idea.

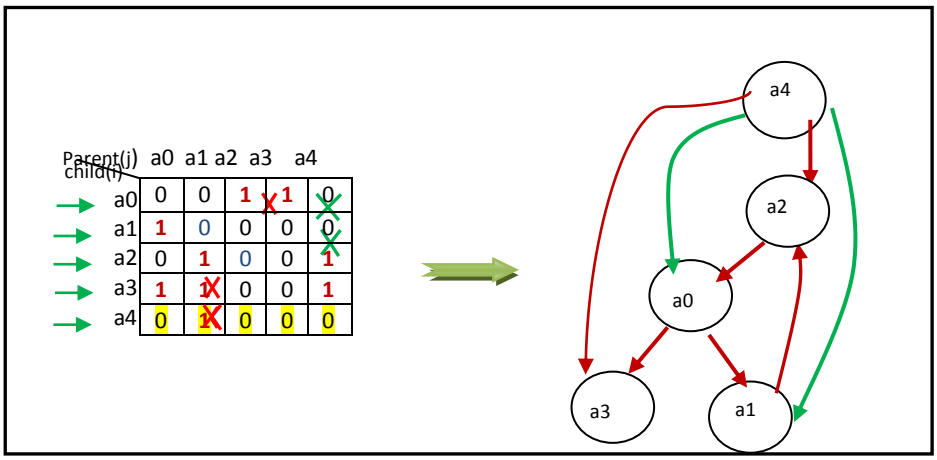


Figure 4.6 (c): Each node has 2 parents (class node and other node), except class node.

iii. There is no circular path emanating from any node. If circularities are detected, they are corrected by selecting one of the links in the cycle at random and removing it, as illustrated in Figure 4.6(d).

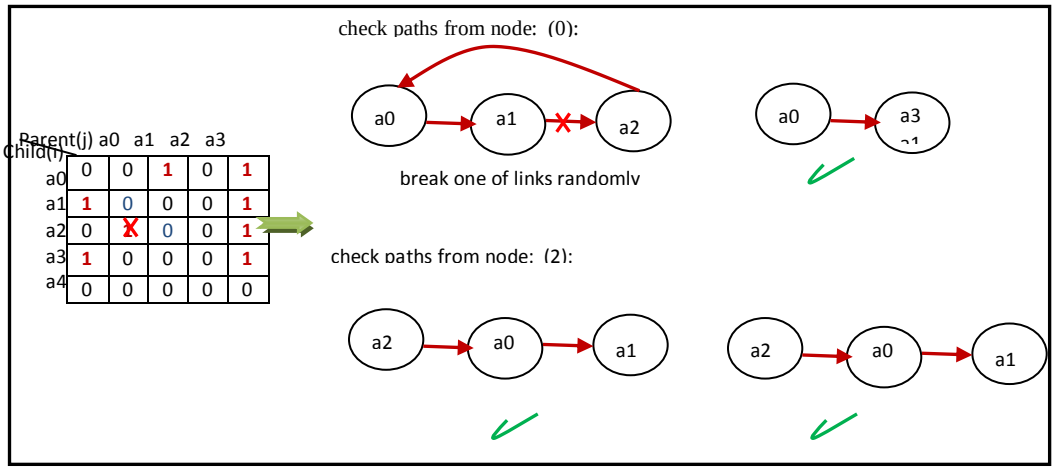


Figure 4.6(d): Testing all paths on adjacency matrix and break.

- Thirdly, given an adjacency matrix representing a valid TAN, it can be converted to a string of bits by arranging it row by row as illustrated in Figure 4.5.

4.3.2 Fitness Function

As well as the representation, there are two more ingredients required to use a GA, namely a *fitness function* and the *operators* required for generating offspring. To generate the offspring, the standard *selection*, *mutation* and *crossover operators* are used together with the above steps for correcting illegal offspring. The fitness function used in this algorithm is the expected cost (that described in Chapter 3, equation (3.3)), it can be expressed as:

$$Fitness\ function = \sum_{j=1}^{k} Cost(i,j) \quad P(j|x) \qquad (4.5)$$

Where, k is the number of classes; P(j|x) represents the probability estimation of classifying the instance x into class j; and $Cost(i,j)$ is the cost of misclassifying class j; the cost of predicting x to class i when the true class of x is j.

4.3.3 Evolving the populations

Once the fitness is evaluated, the next generation is evolved using crossover and mutation randomly as illustrated in Figure 4.7. This process is repeated 20 times and the TAN with least expected cost is selected, where, this number has been chosen based on our experiments, because the optimal tree will be found before 20 trails.

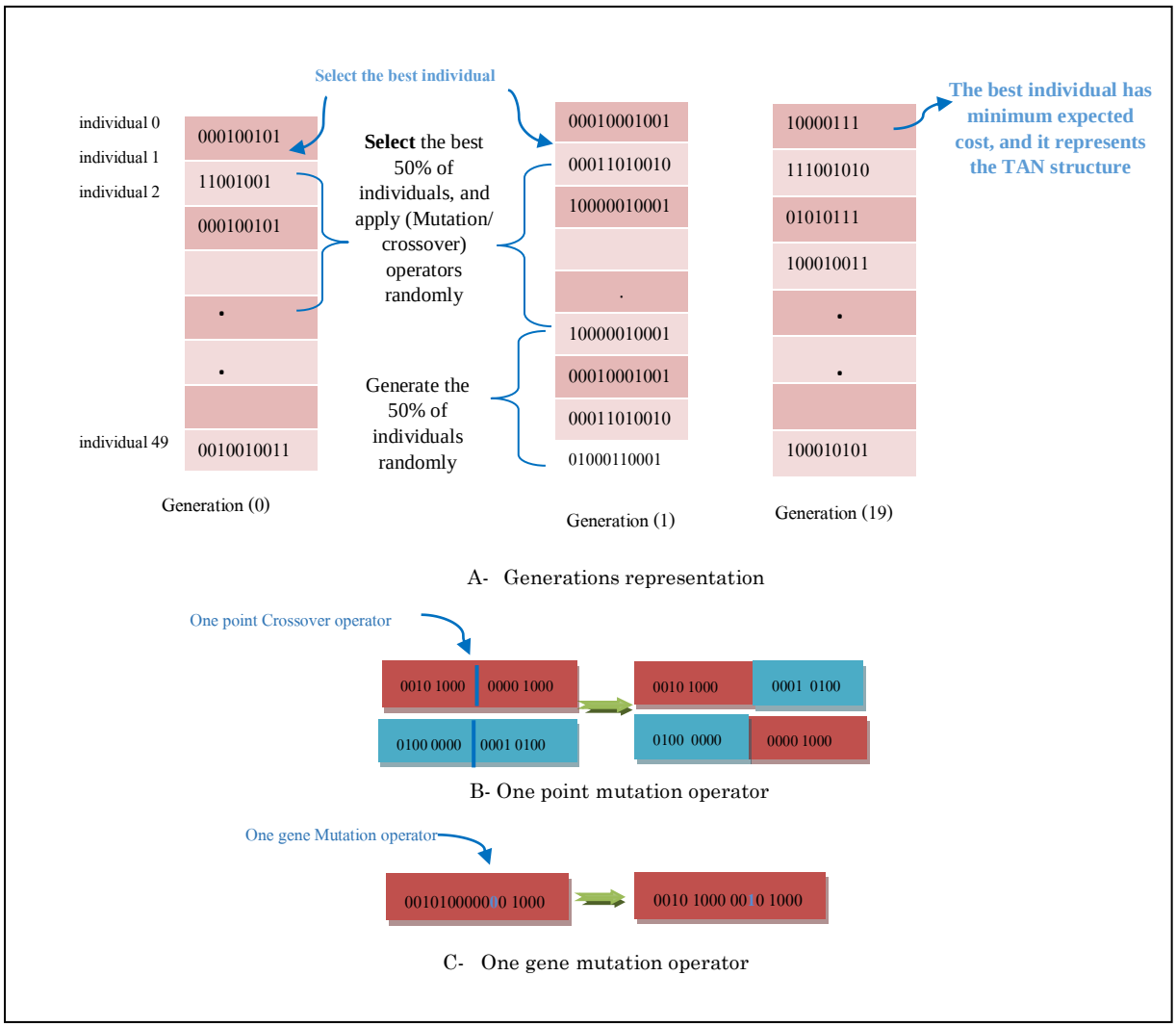


Figure 4.7: Evolving the populations.

These steps lead to the algorithm presented in Figure 4.8. We called this algorithm *CS-BN via GAs*, it has been implemented in *Java netbeans* based on BNs algorithms which available in the *WEKA system (*Witten and Frank, 2005*)*. The implementation is described Appendix C and the code is included in the accompanying CD. An empirical comparison with existing algorithms, such as use of *MetaCost+J48*; *MetaCost+BN* and standard Bayesian networks is presented in Chapter 5.

CS-BN via Genetic algorithm:

1. **Divide** data into 2 sets: 75% training, and 25% testing. And divide training data into 2 sets: 50% sub_training, and 25% sub_testing.
   $Sub_{train} = 50\%, Sub_{test} = 25\%, and\ Testing = 25\%$

// where, Sub_{train} is used for parameters learning, Sub_{test} is used for evaluation fitness function and $Testing$ is used for the final evaluation

2. // Initialize
   ind =1 // ind is the number of individual

   K =1 // K is the number of generation

   P_k is a population of individuals, ind=1To 50. randomly generated individuals[ind]

   Call **Evaluation_Generation$_k$ (P_K, Sub_{train}, Sup_{test})**

3. // other generations k=2 to 20
   do
   {

a. **Select** the first individual (the best) from previous generation $P_{K-1}$and copy it into the current generation P_K

b. **Apply** mutation and cross over randomly on the first half of the previous generation P_{K-1}, ind = 2 to 25 , then **Insert** the new individuals in the current generation P_K

c. **Generate the other individuals in the current generation randomly** P_K , ∀i = 26 to 50

   Call **Evaluation_Generation$_k$ (P_K, Sub_{train}, Sup_{test})**

   K=K+ 1; // next generation
   } **While** (K <= 20)

4. **//Final,** getting the fitness TAN(fitness individual) from the last generation P_{20}.

5. **Evaluate** the best TAN from step 4, by using $Testing$ set, to get the results (accuracy, and cost).

Evaluation generation function:

Evaluation_Generation$_k$ (P_K, D1, D2)
{
ind = 1 ∀ ind = 1 to 50
do
{
Step 1: Check the individual[ind]

IF (ind does not followed TAN's Rule)
Then change the individual[ind] randomly by breaking the circularity in the TAN.

Else continue

Step 2: Build TAN_{ind} structure from the individual[ind]

Step 3: Learn parameters of TAN_{ind} using$D1$

Step 4: Evaluate the TAN_{ind} using$D2$
Compute Fitness function.
Where,
Fitness function = error costs
ind=ind+ 1;
} **While** (ind<= 50)

P_K = Sort the current generation P_K according to the fitness function; using Ascending sort for all individuals in population P_K

Return (P_K).
}

Figure 4.8: CS-BN algorithm using Genetic algorithms.

The main steps of this algorithm are summarised as follows:

Step 1: Splitting data: Randomly divide the dataset into 3 parts:

Sub-training = 50% used for parameter learning.
Sub-testing= 25% used for evaluation the fitness function.
Testing = 25% used for final evaluation (evaluate on the best individual or TAN structure).

Step 2: Randomly create the first generation: The initial population is generated randomly, which is comprised of individuals with random links between attributes(nodes).

Step 3: Check that individuals represent valid TANs: Checks that there are no circular paths, where each node should has just one parent, and the class label is the main parent for all nodes (as illustrated in Figures 4.6).

Step 4: Create TAN structures: After checking each individual follows the TAN's rules, each individual is converted to a TAN structure (as illustrated in Figure 4.5).

Step 5: Learn parameters: After getting the structure for each individual, 25% of the sub-training data is used to learn the parameters for each of the 50 TANs in the population. This is done by using the simple estimator (Freidman et al., 1997) given in equation (2.17) that was described in Section 2.4.3.

Step 6: Evaluation stage: The 25% of the sub-testing data is used to evaluate the fitness function for each TAN structure, where, the fitness function that represents expected misclassification costs for each structure.

Step 7: Get the next generation: The next generation is initialised as follows:

- **individual[0]** is filled with the best individual that has minimum cost in the previous generation which is the first individual in the previous generation as illustrated in Figure 4.7.
- **individual[1,..,25]** are selected from the best individuals in the previous generation (from individual 2 to individual 25) after using the *mutation* and *crossover operators* from the previous generation, as shown in Figure 4.7.

- **individuals[26,..,49]** are generated randomly, as illustrated in Figure 4.7.

Step 8: Get the fitness Bayesian tree: After repeating the whole procedure 20 times, the best TAN structure is obtained with minimum expected costs from the last generation.

Step 9: Evaluating the fitness structure: Finally, the TAN is evaluated using the 25% testing data.

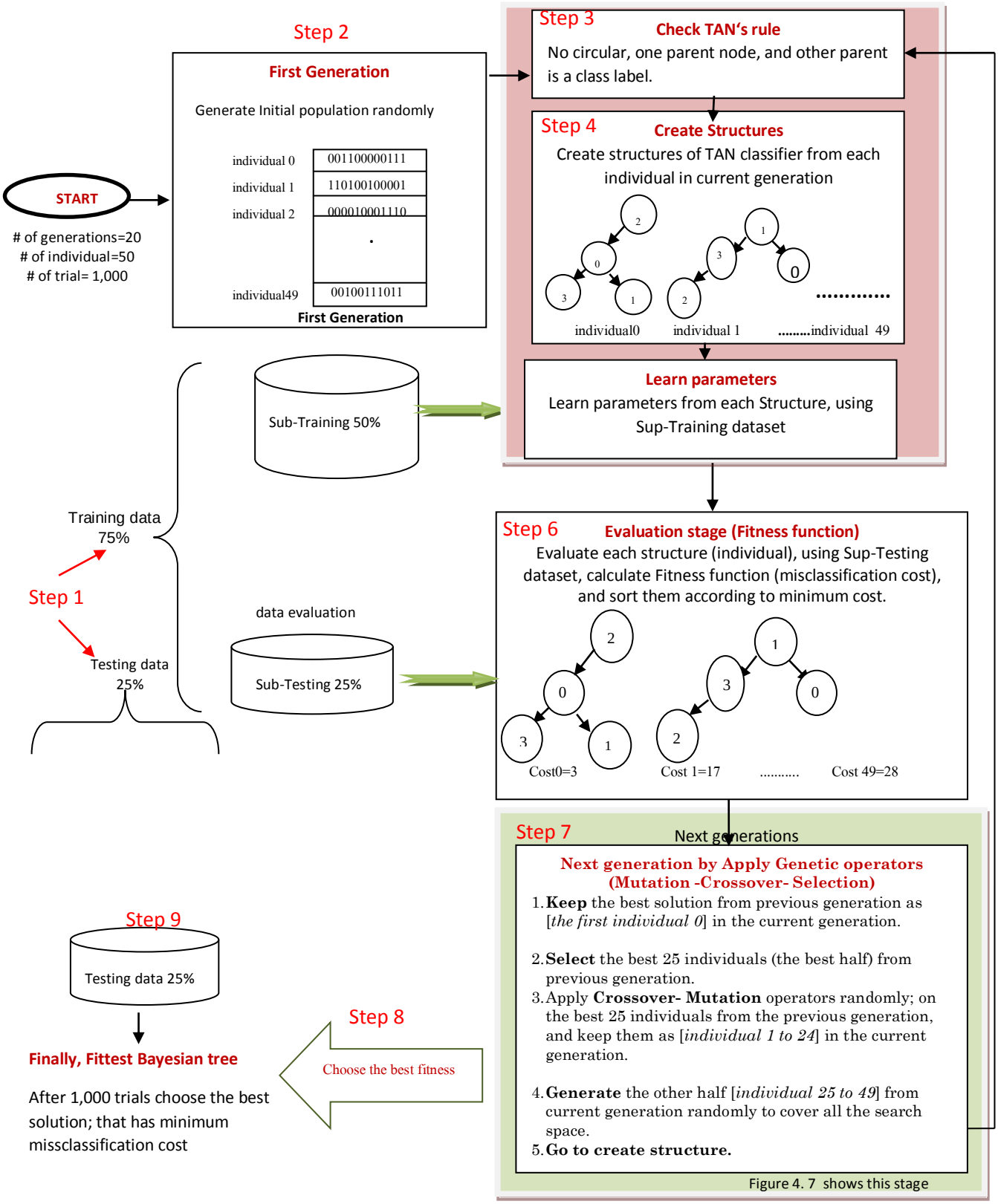


Figure 4. 9: Nine steps to illustrate the main idea of CS-BN via GAs.

4.4 Summary

In this chapter, three new algorithms were developed and presented:

- A cost-sensitive Bayesian network algorithm via the sampling approach, which is based on indirect methods as described in Chapter 3, Section 3.2.1.2.
- A cost-sensitive Bayesian network algorithm via the amending approach, which is based on direct methods as described in Chapter 3, Section 3.2.1.1.
- A cost-sensitive Bayesian network algorithm via Genetic algorithms, which is based on optimizing methods as described in Chapter 3, Section 3.2.1.3.

In this chapter, each algorithm is illustrated in *pseudocode*, and figures, and then, each algorithm is described in detail and summarised in steps. The algorithms have been implemented in the Java based on existing algorithms that available in WEKA system, with an outline of the classes diagram which are presented in Appendix C.

In the next chapter, an empirical evaluation of the new algorithms for learning cost-sensitive Bayesian networks will be presented including the results obtained through the experimental evaluation.

Chapter 5: An Empirical Evaluation of the New Algorithms for Learning Cost-Sensitive Bayesian Networks

Chapter 4 developed three approaches to learning cost-sensitive Bayesian network which are: (i) cost-sensitive Bayesian networks using a sampling approach, (ii) cost-sensitive Bayesian networks using an amending approach, and (iii) cost-sensitive Bayesian networks using the genetic algorithms. This chapter presents the results of an empirical evaluation in order to examine these algorithms and compare their performance with existing cost-sensitive algorithms, such as *MetaCost+J48,* and *MetaCost+BN*, and with cost-insensitive Bayesian network algorithms such as *Tree Augmented Naive Bayes*. This chapter is organised as follows: Section 5.1 presents the results of an empirical comparison; Section 5.2 provides a discussion of the outcomes of the empirical evaluation; and finally, Section 5.3 presents a summary of the findings of the evaluation.

5.1 Empirical comparison results

As explained before, this research develop new cost-sensitive Bayesian network algorithms that take account of misclassification costs, aim to minimise error costs while maintaining the accuracy. This section utilises the empirical methods to assess the extent to which the proposed methods have achieved this aim.

The algorithms that chosen for comparison include:

- A cost insensitive Bayesian network based on *TAN* (Friedman et al., 1997) to provide a base line comparison with Bayesian networks that do not aim to minimise costs as described in earlier in Chapter 2 Section 2.4.1.3.2.
- A cost-sensitive decision tree learner that uses a meta learner *MetaCost+J48* (Domingos, 1999), that described in Chapter 3, Section 3.2.1.2.5, in Figure 3.6.
- A cost-sensitive Bayesian network learner that uses a meta learner *MetaCost+BN* (Domingos, 1999) as described earlier in Chapter 3 in Section 3.2.1.2.5, in Figure 3.6.

All of these algorithms are implemented in the open source data mining software package WEKA (Hall et al., 2009) and they have been adapted to include misclassification costs in their evaluations.

5.1.1 Datasets

In this research, we applied our experiments to 36 datasets, which are available from the UCI Machine Learning Repository, (Asuncion and Newman, 2007). These datasets have been widely used for benchmarking by many researchers with different methods and come from different domains such as physical, medical, and social sciences,…, etc, and have different characteristics as summarised in Table 5.1. Where, Bayesian networks algorithm deal with just nominal attributes and if the attributes are continues which have no pure intervals such as an age attribute, then, Bayesian network algorithm uses a supervised discretization filter to discretize those attributes to nominal attributes as a pre-processing step (Fayyed and lrani, 1993) then deal with the nominal attributes. Figure 5.1 shows that continuous attributes can be cut into many cutting interval points according to class label yes, and no.

64	65	68	69	70	71	72	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No Yes	Yes Yes	No	Yes	Yes	No

Figure 5.1: Discretising data (Fayyed and lrani, 1993).

Then, the frequent nominal values is used to calculate the MDL equation during learning structure (that described in Chapter 2, equation (2.11)), and uses the frequent of nominal values in CPT during learning parameters (that described in Chapter 2, equation (2.17)).

Number of data	Dataset	Class distribution	Instances	Attributes	Type of attributes
1	Adult	(76 : 24)	48842= (37155,11687)	14	5 continuous
2	Australian Credit Approval	(56 : 44)	690=(383, 307)	15	5 continuous
3	Bank	(54 : 46)	600=(362, 274)	11	2 continuous
4	Breast Cancer	(70 : 30)	286=(201,85)	9	All nominal
5	Bupa liver disorder	(58 : 42)	345=(200, 145)	7	6 continuous
6	Cars	(73 : 27)	406=(285, 107)	8	All continuous
7	Cleveland disease	(54 : 46)	303=(165,138)	13	5 continuous
8	Crx	(56 : 44)	689=(382,307)	16	6 continuous
9	Cylinder Band	(58 : 42)	540 =(312,228)	39	17 continuous
10	Diabetes	(65 : 35)	768=(500,268)	8	7 continuous
11	German credit	(70 : 30)	1000=(700,300)	20	7 continuous
12	Gymexamg	(70 : 30)	2500=(1755,745)	20	11 continuous
13	Haberman	(74 : 26)	306=(225,81)	3	2 continuous
14	Hepaties	(97 : 23)	155=(32, 123)	19	6 continuous
15	Horse Colic	(63 : 37)	368=(214,152)	22	14 continuous
16	Horse	(66:34)	370=(215,153)	28	8 continuous
17	Hoslem	(78:22)	189=(147,42)	14	13 continuous
18	Hypo	(95 : 5)	3163=(3012,151)	25	7 continuous
19	IonoSphere	(64 : 36)	351=(225,126)	34	23 continuous
20	kr-vs-kp	(52 : 48)	3196=(1669,1527)	36	All nominal
21	Labor	(65 : 35)	57=(37,20)	16	8 continuous
22	Monks	(50 : 50)	556=(278,278)	7	All nominal
23	Mushroom	(52 : 48)	8124=(4208,3916)	21	All nominal
24	Musk	(52 : 48)	476=(207,269)	168	166 continuous
25	pima_diabetes	(57 : 43)	768=(500,268)	8	All continuous
26	Sick	(94 : 6)	2800=(171, 2629)	29	7 continuous
27	Sonar	(53 : 47)	280=(111,97)	60	All continuous
28	Spambase	(61 : 39)	4601=(2788,1813)	57	All continuous
29	SPECT Heart	(59 : 41)	267=(157,110)	22	All nominal
30	Statlog Heart	(56 : 44)	270=(150,120)	13	All continuous
31	Supermarket	(64 : 36)	4627=(2948,1679)	216	All nominal
32	Tic-Tac-Toe	(65 : 35)	958=(626,332)	9	All nominal
33	Unbalanced	(99 : 1)	856=(844,12)	32	All continuous
34	Vote	(61 : 39)	435=(267,168)	16	All nominal
35	Weather	(64 : 36)	14=(9,5)	5	All continuous
36	Wisconsin Cancer	(66 : 34)	699=(458,241)	10	All continuous

Table 5.1: The main characteristics of datasets used in the comparisons

Experimentally, in this research we use binary classification datasets (i.e. positive and negative class), because, in two class problems it is easier to analysis the misclassification errors and see the differences between correctly classified and incorrectly classified instances, because the cost is opposite to each other. Also, most of research have carry on two class problems (Zadrozny et al., 2003b; Margineantu and Dietterich, 2003), thus, in our experiments we used datasets with two class label.

In addition, in our experiments, we use a wide range of misclassification costs where the cost matrix adopts 16 cost ratios for class1 : class2 as [4:1,4:2,4:3,4:4, 3:1,.., 1:4]. For example, Table 5.2 illustrates that the cost of misclassifying a class C1 as C2 is 4 while misclassification class C2 as C1 is 1.

Predicted class	Actual class	
	C1	C2
C1	0	1
C2	4	0

Table 5.2: Cost matrix of two class labels C1=4, C2=1

The evaluation is carried out using the three methods developed in this thesis: (i) cost-sensitive Bayesian networks via sampling approach based on indirect methods, (ii) cost-sensitive Bayesian networks via amending an existing algorithm based on direct methods, and (iii) cost-sensitive Bayesian networks via Genetic algorithms.

5.1.2 Experiment methodology

The experiment methodology that is used in our research is shown in Figure 5.2. In our experiment methodology, all experiments are repeated with 10 random trials and the results report the averages together with the standard errors.

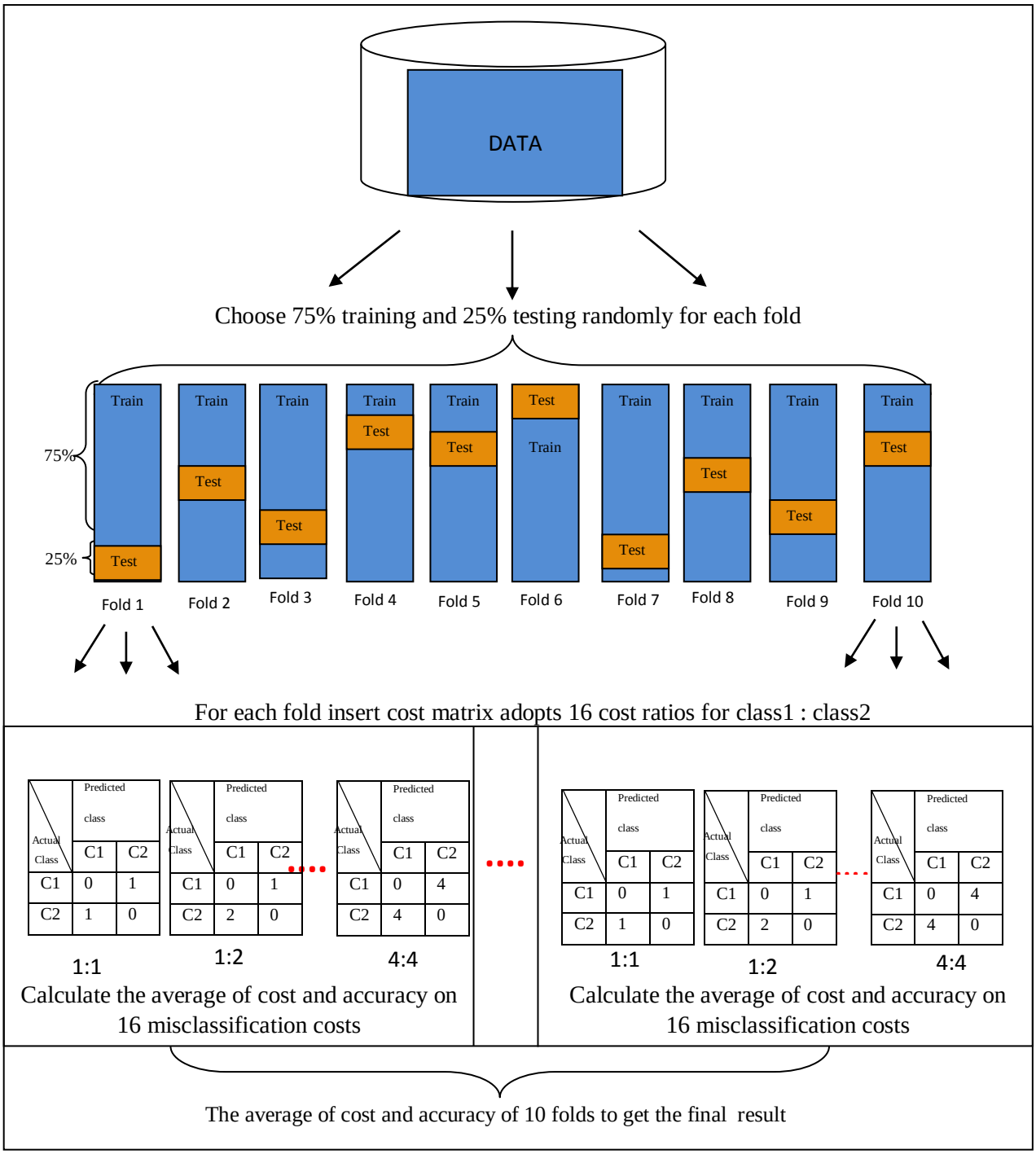


Figure 5.2: The experiment methodology

The final results will be the average cost to classify testing examples, which represent the expected costs of error examples in the testing set. Also, the final result will include the average of the percentage accuracy; this assesses how many of the examples in the testing set have been classified correctly. This methodology has been used for all experiments described in this thesis. In our experiments, we used the algorithms which are available in WEKA open source (Hall et al., 2009), then we write our algorithms with Java language, where these algorithms are implemented with (class implementation) which are illustrated in Appendix C. The following subsections present the results from each of the three approaches developed.

5.1.3 Experiments

In this section we evaluate our three algorithms that described in Chapter 4. Table 5.3 summarises the results of three experiments, which compares our proposed algorithms with the existing algorithms: (1) Original Bayes network (which is their implementation of TAN (Friedman et.al, 1997), version 8). (2) *MetaCost+J48* as the base classifier (which is their implementation of C4.5 version 8) (Domingos, 1999), and (3) *MetaCost+TAN*. Where, Table 5.3 displays the results in form mean squared error ± Standard errors observed for the algorithms. It presents the results for each of the 36 datasets and highlights the result with the lowest cost for each dataset. Figure 5.3 presents the expected costs when each algorithm is applied on the datasets in the form of bar charts, and Figure 5.4 presents the accuracy across different datasets. To make the comparisons in Table 5.3 more easy, three font colours have been used, blue font to determine the first winner, red for the second winner and green for the third winner. Also, we used bold font to determine the lowest cost and highest accuracy for each dataset. Where, 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.

Dataset	CS-BN via Genetic algorithm		CS-BN via sampling approach		CS-BN via amending approach		MetaCost+J48		MetaCost+BN		Original BN	
	Cost	Accuracy	Cost	Accuracy	Cost	Accuracy	Cost	Accuracy	Cost	Accuracy	Cost	Accuracy
Adult	3450.9 ± 15.4	84.01 ± 0.17	3618.8 ± 25.5	79.39 ± 0.08	**3353.7 ± 18.2**	80.05 ± 0.11	3781.6 ± 25.8	81.86 ± 0.13	3622.2 ± 28.66	79.57 ± 0.12	4581.8 ± 23.0	86.11 ± 0.07
Australian Credit	**43.6 ± 3.76**	**90.18 ± 0.77**	43.8 ± 3.32	84.56 ± 0.73	43.8 ± 3.52	84.2 ± 0.78	45.7 ± 3.56	85.74 ± 0.78	54.6 ± 2.49	81.36 ± 0.69	67.1 ± 2.8	84.79 ± 0.4
Bank	72.4 ± 4.99	**81.49 ± 0.82**	**70.8 ± 3.15**	58.85 ± 1.57	71.6 ± 2.9	59.32 ± 1.16	79.5 ± 4.4	55.2 ± 2.42	73.1 ± 2.45	57.5 ± 1.44	111.0 ± 3.13	72.03 ± 1.05
Breast Cancer	**36.4 ± 1.33**	**81.0 ± 0.48**	49.5 ± 2.0	46.0 ± 1.8	55.8 ± 2.62	55.0 ± 1.77	52.9 ± 2.68	61.29 ± 0.89	52.8 ± 2.64	54.14 ± 1.9	58.8 ± 3.13	71.29 ± 1.72
Bupa liver disorder	51.8 ± 1.8	46.51 ± 0.72	50.1 ± 0.1	42.09 ± 0.23	**50.0 ± 0.0**	41.86 ± 0.0	56.0 ± 2.78	58.6 ± 1.51	51.8 ± 1.8	41.51 ± 0.35	133.1 ± 7.27	57.91 ± 0.23
Cars	**0.0 ± 0.0**	**100.0 ± 0.0**	0.9 ± 0.6	99.3 ± 0.4	**0.0 ± 0.0**	100.0 ± 0.0	**0.0 ± 0.0**	100.0 ± 0.0	0.4 ± 0.4	99.88 ± 0.12	0.4 ± 0.4	99.88 ± 0.12
Cleveland disease	**24.0 ± 1.65**	**87.2 ± 1.0**	27.2 ± 1.15	78.93 ± 0.76	29.4 ± 2.28	79.6 ± 1.29	29.7 ± 1.51	74.4 ± 0.88	29.5 ± 2.34	77.47 ± 1.08	32.4 ± 1.56	82.4 ± 0.59
Crx	48.2 ± 2.9	**89.41 ± 0.63**	47.0 ± 4.04	83.55 ± 0.93	50.8 ± 3.61	84.14 ± 1.05	**38.5 ± 3.2**	85.56 ± 0.76	51.7 ± 3.4	81.12 ± 1.12	62.2 ± 3.2	86.98 ± 0.85
Cylinder Band	**59.9 ± 3.09**	**81.49 ± 0.9**	87.2 ± 5.15	68.73 ± 1.22	92.8 ± 4.57	73.51 ± 0.85	77.0 ± 0.0	42.54 ± 0.0	93.7 ± 5.44	71.49 ± 1.13	98.5 ± 5.12	74.85 ± 1.04
Diabetes	90.2 ± 3.78	61.41 ± 1.28	87.4 ± 4.69	68.53 ± 1.01	**85.9 ± 3.33**	66.49 ± 0.75	100.0 ± 4.59	70.1 ± 0.73	88.3 ± 4.19	68.06 ± 0.89	126.1 ± 5.57	76.07 ± 0.69
German credit	138.0 ± 4.61	**79.0 ± 0.69**	137.6 ± 3.53	55.4 ± 0.88	**128.9 ± 4.66**	67.76 ± 0.98	157.2 ± 6.41	64.24 ± 1.34	138.2 ± 5.78	66.32 ± 0.87	187.1 ± 6.88	72.92 ± 0.98
Gymexamg	**438.0 ± 0.0**	29.35 ± 0.0	**438.0 ± 0.0**	29.35 ± 0.0	**438.0 ± 0.0**	29.35 ± 0.0	566.4 ± 11.6	46.39 ± 0.87	**438.0 ± 0.0**	29.35 ± 0.0	728.0 ± 0.0	70.65 ± 0.0
Haberman	53.6 ± 1.18	54.93 ± 2.57	52.3 ± 1.51	33.87 ± 1.48	**51.1 ± 2.2**	51.87 ± 2.19	51.3 ± 2.72	63.6 ± 1.57	**57.8 ± 1.74**	52.53 ± 1.34	73.3 ± 1.72	71.07 ± 0.87
Hepaties	**6.9 ± 1.2**	**91.54 ± 1.08**	13.2 ± 1.32	78.46 ± 1.63	12.8 ± 1.7	82.56 ± 1.66	18.7 ± 1.94	78.21 ± 1.76	14.0 ± 1.44	83.33 ± 1.59	15.0 ± 1.69	84.62 ± 1.62
Horse Colic	**27.7 ± 1.44**	**85.54 ± 0.97**	42.9 ± 2.06	71.3 ± 1.75	44.7 ± 2.74	76.52 ± 1.43	45.8 ± 2.8	79.57 ± 1.2	45.3 ± 2.68	73.26 ± 1.54	45.4 ± 2.77	80.65 ± 1.19
Horse	**29.0 ± 3.12**	**82.97 ± 1.37**	39.8 ± 1.9	72.09 ± 1.24	46.4 ± 3.69	72.09 ± 1.6	61.0 ± 0.0	32.97 ± 0.0	51.6 ± 3.47	63.41 ± 1.95	50.3 ± 3.4	76.37 ± 1.2
Hoslem	**0.0 ± 0.0**	**100.0 ± 0.0**	**0.0 ± 0.0**	**100.0 ± 0.0**	**0.0 ± 0.0**	**100.0 ± 0.0**	**0.0 ± 0.0**	**100.0 ± 0.0**	1.5 ± 1.07	98.04 ± 1.31	0.0 ± 0.0	100.0 ± 0.0
Hypo	**9.6 ± 0.9**	**99.23 ± 0.09**	18.3 ± 2.61	98.31 ± 0.2	18.1 ± 1.38	98.19 ± 0.17	10.6 ± 1.59	99.41 ± 0.08	18.3 ± 2.61	98.31 ± 0.2	21.8 ± 2.69	97.94 ± 0.2
IonoSphere	**12.6 ± 1.22**	94.83 ± 0.57	20.8 ± 2.05	89.89 ± 1.03	24.3 ± 2.1	89.66 ± 0.82	28.6 ± 2.58	86.09 ± 1.47	27.9 ± 2.58	88.97 ± 0.96	26.9 ± 3.45	89.77 ± 1.12
kr-vs-kp	**106.6 ± 4.66**	**93.88 ± 0.34**	134.1 ± 5.14	84.68 ± 0.45	146.9 ± 2.73	84.2 ± 0.18	139.0 ± 5.83	92.29 ± 0.24	171.0 ± 4.76	83.12 ± 0.41	171.0 ± 4.76	83.12 ± 0.41
Labor	**1.3 ± 0.6**	**97.14 ± 1.17**	4.3 ± 0.72	84.29 ± 2.56	4.1 ± 1.04	85.71 ± 2.61	5.6 ± 0.64	81.43 ± 1.9	4.6 ± 1.02	84.29 ± 2.56	5.8 ± 1.58	86.43 ± 2.49
Monks	**0.0 ± 0.0**	**100.0 ± 0.0**	**0.0 ± 0.0**	**100.0 ± 0.0**	**0.0 ± 0.0**	**100.0 ± 0.0**	32.9 ± 1.46	76.16 ± 1.06	0.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	100.0 ± 0.0
Mushroom	**0.0 ± 0.0**	**100.0 ± 0.0**	4.9 ± 1.91	99.94 ± 0.03	1.6 ± 0.88	99.98 ± 0.01	**0.0 ± 0.0**	**100.0 ± 0.0**	3.2 ± 1.16	99.96 ± 0.01	2.8 ± 1.2	99.97 ± 0.01
Musk	23.4 ± 3.67	93.08 ± 1.02	22.8 ± 2.43	88.46 ± 1.32	22.7 ± 3.25	92.91 ± 0.85	**19.7 ± 8.31**	**93.42 ± 2.35**	33.3 ± 5.47	86.67 ± 1.8	24.2 ± 3.89	91.88 ± 1.18
Pima diabetes	**90.9 ± 3.1**	68.12 ± 0.93	94.3 ± 4.06	58.17 ± 1.86	91.1 ± 3.18	67.07 ± 0.97	101.4 ± 4.73	70.16 ± 1.21	110.6 ± 3.62	75.55 ± 0.72	122.3 ± 4.25	75.24 ± 0.84
sick	35.6 ± 2.1	97.6 ± 0.11	41.7 ± 3.9	96.94 ± 0.17	41.7 ± 2.86	96.9 ± 0.15	**25.6 ± 2.45**	**98.09 ± 0.13**	41.7 ± 3.9	96.94 ± 0.17	39.4 ± 2.6	97.58 ± 0.06
Sonar	**25.8 ± 2.31**	**80.38 ± 1.34**	29.5 ± 2.22	65.77 ± 1.46	32.4 ± 2.07	70.0 ± 1.12	32.8 ± 3.72	66.92 ± 2.72	32.8 ± 2.17	66.92 ± 2.16	35.5 ± 2.11	74.42 ± 1.77
Spambase	214.9 ± 6.84	**93.12 ± 0.2**	**172.8 ± 7.43**	92.76 ± 0.16	197.9 ± 8.09	92.9 ± 0.15	182.8 ± 9.24	91.29 ± 0.28	234.4 ± 6.44	91.95 ± 0.17	230.5 ± 8.55	92.27 ± 0.2
SPECT Heart	**35.8 ± 1.73**	**76.21 ± 1.08**	42.2 ± 2.77	56.97 ± 2.5	37.0 ± 1.83	64.85 ± 1.35	40.3 ± 2.96	64.39 ± 1.52	39.2 ± 2.04	63.79 ± 1.18	53.5 ± 3.68	68.48 ± 1.55
Statlog Heart	24.9 ± 1.77	**85.91 ± 0.96**	23.4 ± 1.93	79.55 ± 1.89	**24.5 ± 2.09**	82.42 ± 1.2	24.6 ± 2.9	77.27 ± 1.88	26.4 ± 1.13	74.09 ± 0.94	26.7 ± 1.41	84.09 ± 1.04
Supermarket	**727.0 ± 0.0**	36.45 ± 0.0	**727.0 ± 0.0**	36.45 ± 0.0	**727.0 ± 0.0**	36.45 ± 0.0	**727.0 ± 0.0**	36.45 ± 0.0	**727.0 ± 0.0**	36.45 ± 0.0	1668.0 ± 0.0	63.55 ± 0.0
Tic-Tac-Toe	127.7 ± 3.59	**80.34 ± 0.48**	124.6 ± 2.05	52.92 ± 0.59	110.3 ± 2.73	64.19 ± 0.49	**98.6 ± 5.63**	79.83 ± 0.9	130.1 ± 4.03	57.46 ± 0.92	166.2 ± 4.22	77.25 ± 0.62
Unbalanced	8.0 ± 0.0	**99.05 ± 0.0**	**7 ± 0.35**	97.62 ± 0.17	8.0 ± 0.0	99.05 ± 0.0	8.0 ± 0.0	99.05 ± 0.0	8.0 ± 0.0	99.05 ± 0.0	8.0 ± 0.0	99.05 ± 0.0
Vote	**6.2 ± 1.05**	**96.73 ± 0.32**	11.2 ± 0.94	94.58 ± 0.27	10.9 ± 1.08	95.42 ± 0.45	11.3 ± 1.88	94.49 ± 0.63	15.7 ± 1.61	92.34 ± 0.76	15.7 ± 1.87	93.46 ± 0.44
Weather	**0.0 ± 0.0**	**100.0 ± 0.0**	2.4 ± 0.37	40.0 ± 4.44	2.4 ± 0.37	40.0 ± 4.44	2.4 ± 0.37	40.0 ± 4.44	2.6 ± 0.58	33.33 ± 7.03	2.6 ± 0.69	63.33 ± 7.78
Wisconsin Cancer	**4.0 ± 0.83**	**98.02 ± 0.29**	5.5 ± 0.87	97.15 ± 0.36	5.4 ± 0.96	97.38 ± 0.34	11.8 ± 1.39	95.41 ± 0.49	7.0 ± 1.34	97.15 ± 0.34	8.4 ± 1.27	97.21 ± 0.38

Table 5.3: Comparison between CS-BN algorithms and existing algorithms

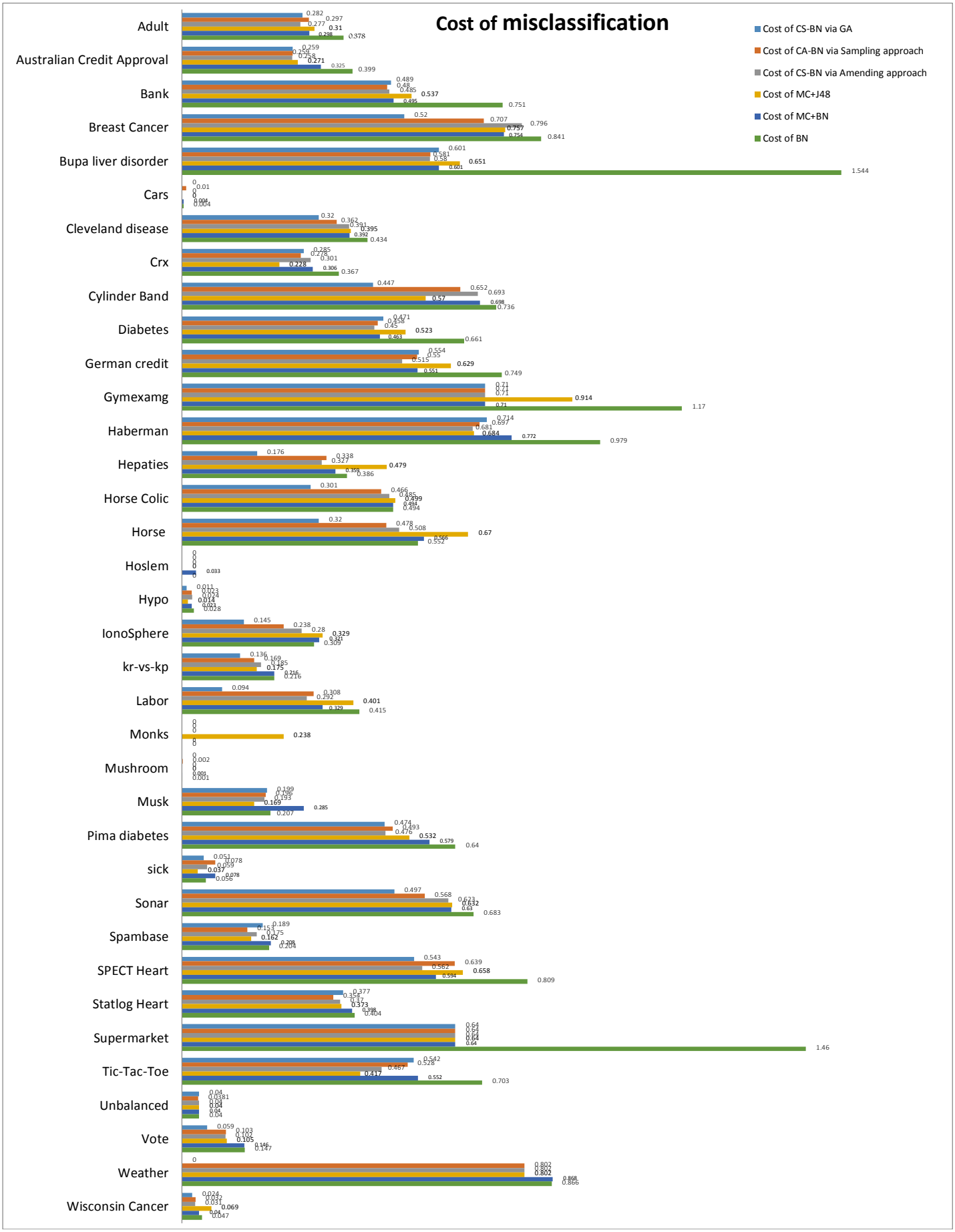


Figure 5.3: Expected cost of CS-BN algorithms and existing algorithms

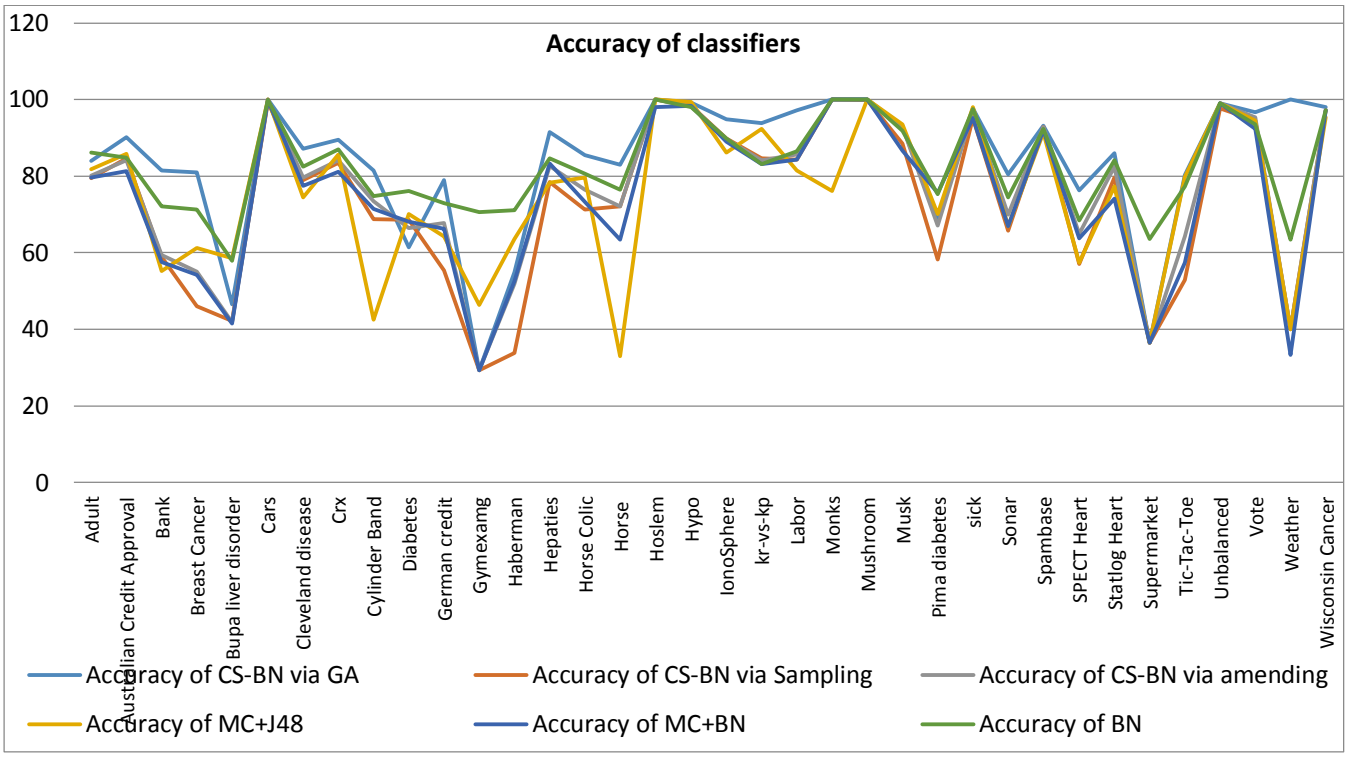


Figure 5.4: Accuracy of CS-BN algorithms and existing algorithms

5.1.3.1 Experiment 1: CS-BN using the sampling approach

In this experiment, we evaluate the CS-BN via sampling approach described in Chapter 4 Section 4.1 that based on changing the data distribution to reflect the cost. These experiments show that:

(i) The numbers of misclassifications of the rare class (often the more expensive class) are always less than the number of misclassifications of frequent class in all datasets. Thus, sampling has the intended effect since it is minimizing the cost by duplicate rare instances and deleting some of frequent instances according to misclassification costs. Thus, this will increase the numbers of misclassifications of the frequent class and decrease the number of misclassifications of the rare class. For example, in the breast cancer dataset, a false positive error means unnecessary treatment; unnecessary worry, while, a false negative error means postponed treatment or failure to treat; death or injury. Figure 5.5 shows how cost-sensitive CS-BN via sampling approach decreases the average number of rare class (FN) in the breast cancer data comparing with existing Bayesian network algorithm as shown in Table 5.4. Thus, it would enable a clinician to review such cases and avoid missing

potential cases of cancer that need treatment. In contrast, when the cost is decreased then the accuracy will be decreased because, sampling approach aims to minimize the costs by decreasing the number of more expensive class FN, even if the number of cheap class is increased FP. Thus, that will decrease the accuracy because some of unimportant instances are misclassified. For example, in Table 5.4, in the first trial, the cost, and accuracy of the CS-BN via sampling will be: Cost= 3*4 + 1* 38=50 and accuracy = 29/ 70 =%41.42

While, the cost and accuracy of the original BN will be: Cost= 3*14 + 1*11 = 67 and accuracy= 45/70 =%64.28

Calculating the cost and accuracy are based on equations (3.1, and 3.2).

CS-BN via Sampling approach				**Original BN**			
No. of rare (FN)	No. of frequent (FP)	Expected cost	Accuracy	No. of rare (FN)	No. of frequent (FP)	Expected Cost	Accuracy
3	38	50	%41.42	14	11	67	%64.28
1	37	41	%45.71	10	4	44	%80
1	45	49	%46.0	14	10	66	%65.71
3	29	41	%54.29	12	9	57	%70
4	35	51	%44.29	15	7	67	%68.57
3	38	50	%41.42	12	6	54	%74.28
3	38	50	%41.42	15	7	67	%68.57
4	35	51	%44.29	9	5	41	%80
3	29	41	%54.29	13	6	58	%72.85
6	33	57	%44.29	15	7	67	%68.57

Table 5.4: The results of CS-BN via sampling and original BN algorithm for the breast cancer dataset.

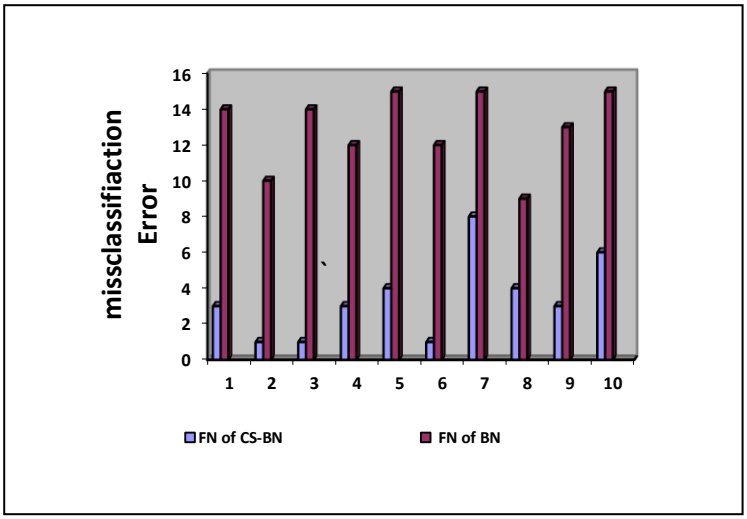
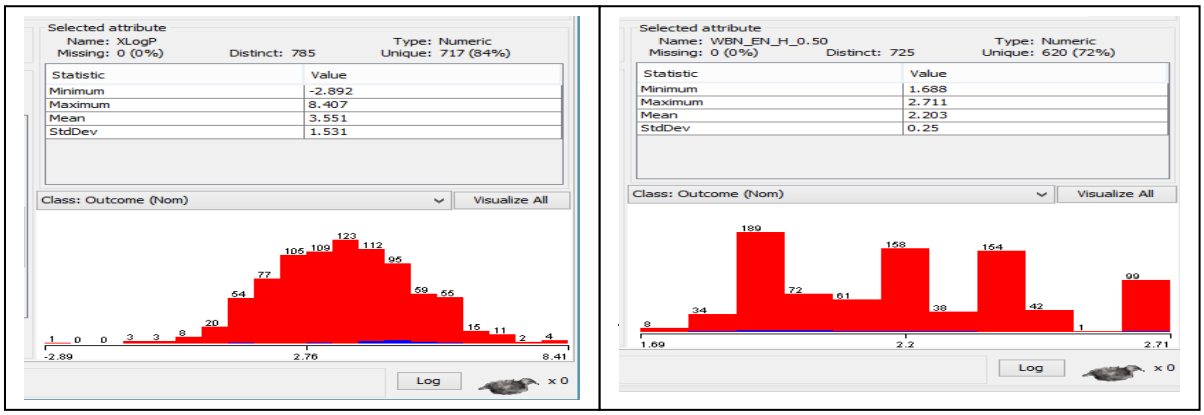


Figure 5.5: Misclassification error if experiment 1 for breast cancer dataset

(ii) Sampling followed by use of the TAN classifier, yields good results on most datasets; especially if the data are very highly skewed towards one class.

Unbalanced dataset has the proportion of rare class at 1% (number of active instances 12) while the frequent class is 99% (number of inactive instances 844), and if the cost of rare class is 8 and the cost of frequent class is 1, then the new distribution will be 11% (Active instances= 96): 89% (inactive instances = 844) instead of 1%: 99% . In particular, increasing the number of rare class examples means increased rare instances, which are very expensive. Therefore, after changing the distribution, the learner will build the Bayesian tree classifier based on the new distribution, and as the result of the experiment, the classifier will classify the new instances and take into consideration the rare instances.

(iii) CS-BN algorithm via sampling apprach works better when the data distribution has the same pattern. This means the data instances are similar and there is little diversity. When instances are very similar, sampling will duplicate similar instances or still have similar instances even when the instances are removed, such as, Hoslem, and monks datasets. In particular, the performance of this method will be very good when the original data distribution has low variation, because all instances are spread out around the centre (mean) of a dataset, and there are few outlier instances, as shown in Figure 5.6(a). However, the performance of this method will not be good when the original data distribution are widespread, because lots of outlier instances are spread out far from the centre of the dataset, and there are lots



(a) Low variation, attribute variables are similar (b) High variation, attribute variables are diverse

Figure 5.6: WEKA a pre-process stage shows the similarity and diversity of attribute variables

For example, in the Cars dataset, all instances have the same pattern, and therefore, sampling approach will work very well in these datasets because it will duplicate some similar instances or remove some of the similar instances. Experimentally, sampling approach changes the data distribution by increasing the instances that

belong to class label when these instances are similar. As a result, the classifier will predict these instances correctly, and thus, this algorithm gives less expected cost compared with original Bayesian network. On the other hand, in these types of data such as *Crx, and Cylinder Band* data distribution, the sampling approach will not work very well because the data distribution are highly varied, and using sampling approach might duplicate some of the unimportant instances, or delete some of the important ones.

(iv) Overall, CS-BN using the sampling approach outperforms *MetaCost+BN*, and the original algorithm in terms of minimising cost in all datasets. In particular, to compute the class probability estimates, the *MetaCost* algorithm uses votes upon which class probabilities are produced by bagging are based on a measure of the variance of BN learner on a particular example. As a result, we find that the classifier that has high variance as shown in Figure 5.6(b), the base learner is less stable in a particular example, as Green (2010) mentioned that *"in MetaCost algorithm the variance is not the same as the class probability"*. *Variance* describes how widely data of BN base learner on a particular instance are spread out about the center of a dataset. The *class probability* is produced by the ensemble, which is the fraction of trained classifiers that predict that particular class (Margineantu, 2000). For example, if a base learner has learned to classify a particular instance that has a true probability of being in class 1 of 60%, each classifier in the ensemble may predict class 1 resulting in a class probability estimate of 100%, where there is 40% belonging to class 2. For this reason, the bagging is not a good choice for estimating class probabilities (Margineantu, 2002; Green, 2010). Therefore, *MetaCost+BN* performs less well on the datasets than other costing sensitive algorithms.

(v) CS-BN via sampling outperforms *MetaCost+DT* algorithm on most of the datasets. *MetaCost+J48* algorithm may work better on some datasets such as Tic-Tac-Toe, Crx, Cylinder Band; when the decision trees obtained with J48 give better results than the original BN in terms of accuracy.

(vi) As shown in Figure 5.4, the accuracy of the CS-BN via sampling approach is similar or slightly less than the accuracy of original BN.

5.1.3.2 Experiment 2: CS-BN using the amending approach

In this experiment, we evaluate the CS-BN via amending approach described in Chapter 4 Section 4.2 where we include misclassification costs into the Bayesian tree (TAN) learning process by changing the formula for *Minimum Description Length* MDL (Rissanen, 1978) to include misclassification costs. In particular, we include costs during both learning structure as described in Section 4.2.1 and learning parameters as described in Section 4.2.2. The main findings from these results are:

(i) The number of misclassifications of the rare class (more expensive) in this approach is always less than the number of misclassifications of rare class in the original TAN algorithm. Therefore, the new algorithm gives a better result in terms of costs compared to the original Bayesian network learning algorithm.

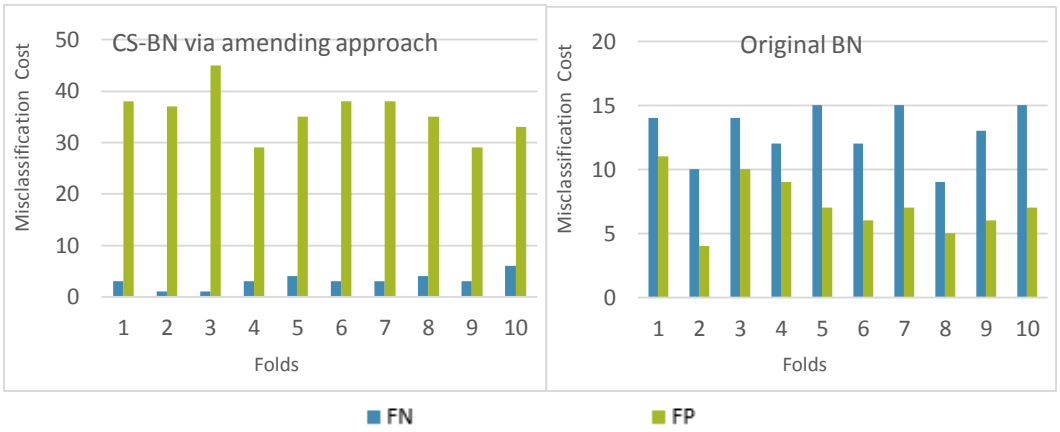


Figure 5.7: Misclassification error if experiment 2 for breast cancer dataset

(ii) In experiment 2, our algorithm works better than *MetaCost+BN* in all datasets for the same reason explained in experiment 1, that the higher the variance, the less accurate the estimate of the conditional probabilities. Where, this experiment gives similar results in some datasets that have the same pattern; similar instances such as Cars, Hoslem, and monks.

(iii) CS-BN via amending outperforms *MetaCost+DT* algorithm on most of the datasets. But *MetaCost+DT* algorithm may work better on some datasets, when, the decision trees obtained with J48 give better results than the original BN in terms of accuracy such as Crx, Cylinder, and Tic-Tac-Toe datasets. In particular, if the cost insensitive decision trees (J48) are better than the existing BN in terms of accuracy, then *MetaCost+j48* is more likely to be better than *MetaCost+BN*, and *CS-BN* via the amending approach, and *MetaCost+BN*.

(iv) As shown in Figure 5.4, the accuracy of the cost-sensitive version is similar or slightly less than the accuracy of the original BN, and the level of sacrifice is not as significant as reported in studies that use similar approaches for learning cost-sensitive decision trees (Jiang et al., 2014).

5.1.3.3 Experiment 3: CS-BN using Genetic algorithms

This section presents an empirical evaluation of CS-BN using genetic algorithm, which is carried out by using 75% of the data for training and 25% for testing. The 75% of training data is further subdivided to two parts: 50% is used for learning the parameters and 25% is used for assessing the fitness function (see Chapter 4, Section 4.3 for more details). These experiments show that:

(i) Overall, CS-BN via Genetic algorithms outperforms *MetaCost+J48,* and *MetaCost+BN* in terms of minimizing cost, while simultaneously increasing accuracy. For example, on the Adult dataset, the average cost and accuracy of our algorithm is 3450.9 and 84% respectively while for *MetaCost+BN* these are 36220.2 and 79.75% respectively. The accuracy of the CS-BN version is better than other classifiers, including the original accuracy based version of TAN, because this algorithm chooses the best cost and accuracy in each trial, then sorts the Bayesian networks according to the best fitness that has minimum cost and maximum accuracy as described in Chapter 4, Section 4.3.

(ii) This algorithm aims to minimise cost according to the fitness function, which uses the misclassification cost just in the evaluation step in the fitness function (expected cost); obviously, it does not include the misclassification cost when creating a BN, but in each trial the algorithm chooses the best BN (that has minimum misclassification cost); where the minimum misclassification cost means minimum of FN and FP together, so the weight given to the accuracy of FN and FP will be similar, not like other cost sensitive algorithms that aim to minimise FN. For example, in breast cancer dataset as shown in Figure 5.8, where it shows the results of misclassification costs between existing algorithms, obtained as an average of 10 random trials.

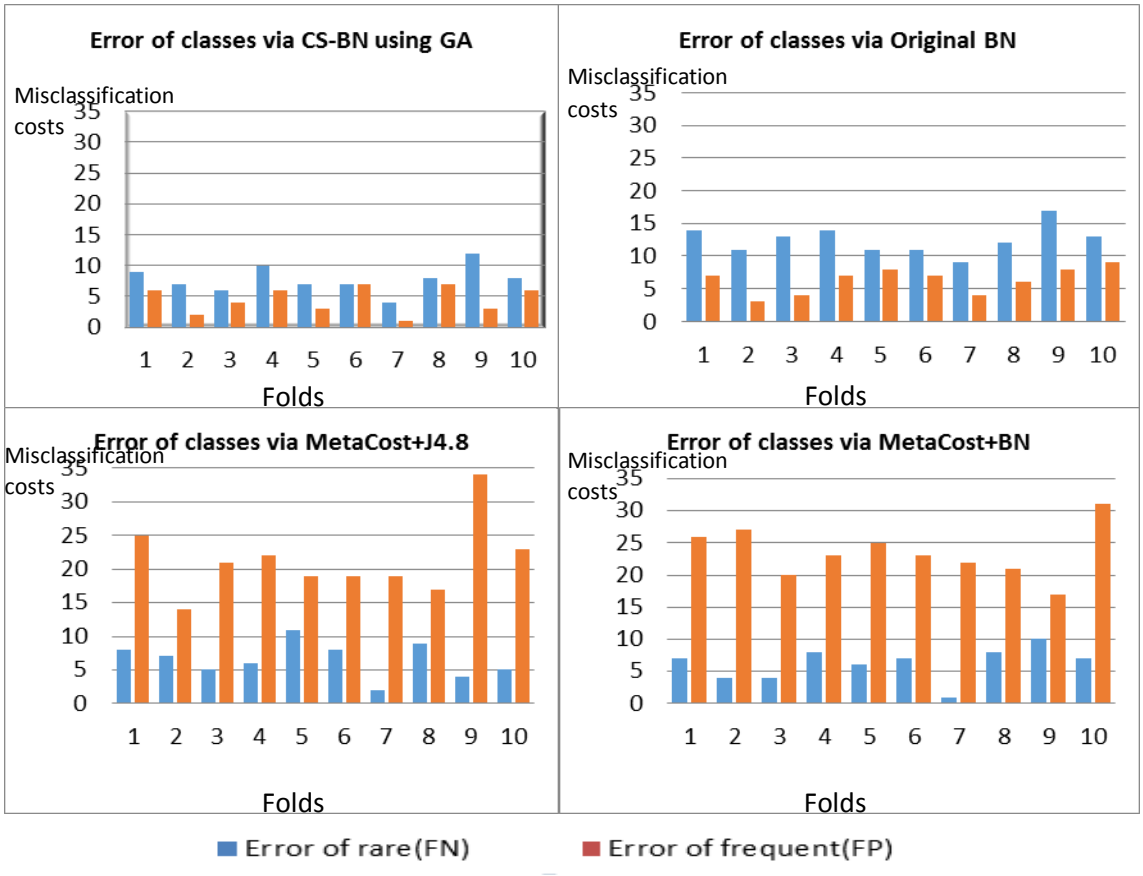


Figure 5.8: CS-BN via GA reduces the number of misclassification error for the Breast Cancer dataset.

(iii) This algorithm will take a long time, when the data has lots of attributes, because this algorithm generates the Bayesian tree randomly, thus, the search space will be very large if the data has lots of attributes, such as the spambase data where the number of attributes is 53. As a result, searching for the fittest TAN will take longer. In contrast, it will work very well if the data has a low number of attributes such as diabetes dataset where the number of attributes is 9. In particular, many researches have suggested feature selection methods to reduce the dimensional of the data (Dash and Liu, 1997; Dash and Liu, 1997). If the data has lots of features in this case one of the feature selection methods should be used to reduce the training time.

(iv) Several authors have reported significant issues with the performance of learning algorithms on imbalanced datasets, and hence as well as the above experiments, we also carried out experiments to examine the performance of algorithms on four imbalanced datasets as cost ratios are increased. Figure 5.9 shows the results, obtained as an average of 10 random trials, and as the cost ratio of misclassification of one class over another is increased from 50 to 400. The results for these datasets

show that our algorithm performs better than the *MetaCost* classifier when the cost of misclassifications is increased.

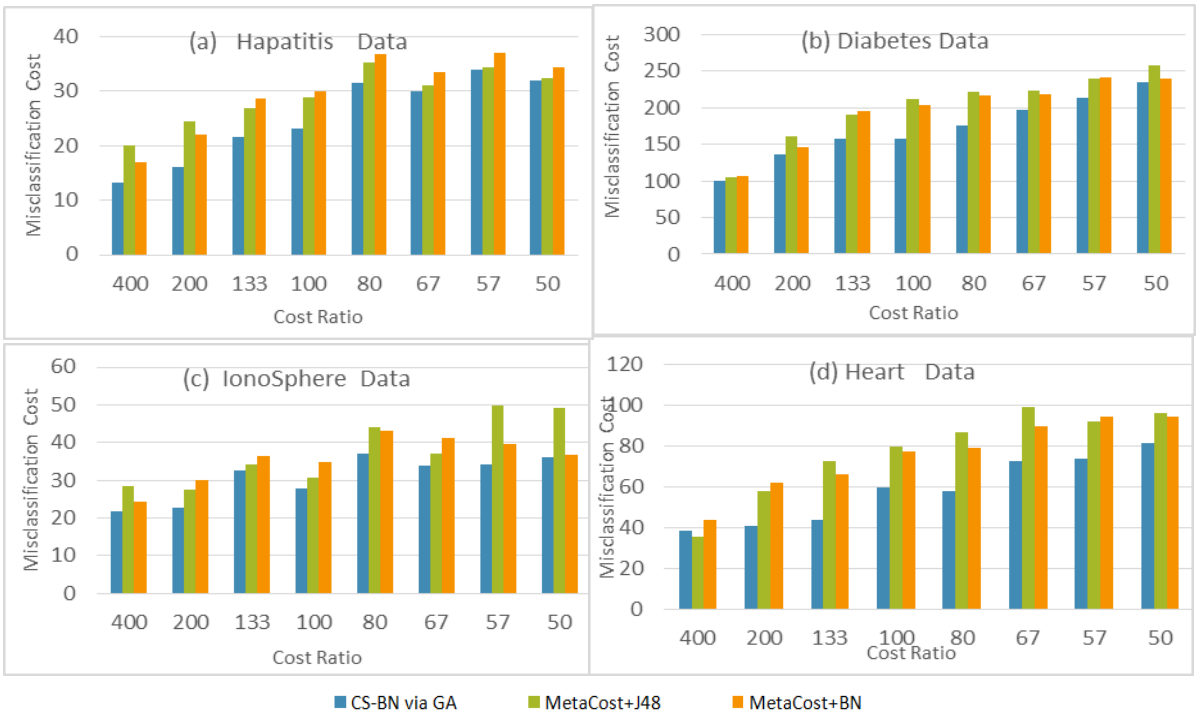


Figure 5. 9: Misclassification costs in the cost ratio range 50 to 400 in CS-BN via GA

5.2 Comparison of the three algorithms developed

As we mentioned in Chapter 1, there are many types of classification algorithms, and there is no such thing as the best algorithm, because it depends on type of data and the pattern of data as well (Wolpert, 1995). To make the comparisons in Table 5.3 more easy, three colours have been used, the blue font to determine the first winner, the red for the second winner and the green for the third winner, where the results show that CS-BN via GA is the best algorithm in term of accuracy and cost. As shown in Table 5.3, CS-BN via GA wins 24 times, then the CS-BN via amending has good results in terms of cost, where it wins 13 times. From the algorithms developed, the CS-BN via sampling approach is the relatively least effective with 8 wins. All algorithms have the same results on 4 datasets that is because they have the greatest results, which are Supermarket, Monks, Hoslem, and Gymexamg. However, in term of accuracy the CS-BN via GA is win for the most of cases; for (26 times), even if the cost was worst.

5.3 Summary

This chapter has presented an empirical evaluation of the three new algorithms developed in Chapter 4. The results show that the algorithms CS-BN via sampling, and CS-BN via amending have a similar pattern of results when compared with existing cost-sensitive algorithms, but CS-BN via GAs has a different pattern of results. For both CS-BN via sampling and via amending, the numbers of misclassifications of the rare class (more expensive) FN are always less than the number of misclassifications of frequent class FP in all datasets. While in CS-BN via GAs the number of FN are nearly similar to FP as shown in Figure 5.8.

All three new algorithms outperform the cost-insensitive Bayesian network algorithm for all the datasets in terms of minimising cost. They are also better than the existing cost-sensitive Bayesian networks algorithm, *MetaCost+BN*, in terms of cost. On the other hand, when the cost in-sensitive decision tree learning algorithm, J48 is better than the existing BN in terms of accuracy, then MetaCost+j48 is more likely to be better than *MetaCost+BN* and our algorithms, which happens on four datasets in average; though it is worth noting that our algorithms outperform MetaCost+J48 on 32 out of the 36 datasets. In term of misclassification cost, the best algorithm is CS-BN via GAs, then via amending then via sampling approach. The accuracy of both algorithms CS-BN via sampling, and via amending is similar though slightly less than the original accuracy based version of TAN, while CS-BN via GAs performs better results than all other algorithms in term of accuracy and cost as well.

In the next chapter, a conclusion that summarises the achievements made in this research and how the research objectives have been addressed will be presented, with the results obtained through the experimental evaluations.

Chapter 6: Conclusions and Future Work

This chapter presents the conclusions of this study: Section 6.1 presents a summary of the context and motivation, Section 6.2 revisits the objectives and reflects critically on the extent to which the objectives are met, and Section 6.3 presents limitations and possible areas for future work.

Through the past decade, the problem of developing algorithms that can induce cost-sensitive classifiers has become a significant challenge. Thus, cost-sensitive learning algorithms have received increasing attention in most real world applications. The majority of research studies on cost-sensitive learning algorithms have focused on the induction of decision trees with either direct amendments to existing algorithms or the use of indirect methods such as bagging and boosting (Lomax and Vadera, 2013). Bayesian networks have been shown to be an effective classifier with a number of useful characteristics, and hence, an obvious question stems from whether or not Bayesian networks can result in classifiers that perform better when it comes to minimising costs of misclassification?. However, existing Bayesian network algorithms that are designed to minimise misclassification errors do not take misclassification costs into consideration. As a consequence, this study has explored whether or not it is possible to develop cost-sensitive Bayesian networks. Overall, three algorithms were developed by analogy with the strategies used for developing cost-sensitive decision trees:

(i) Cost-sensitive Bayesian networks via a sampling approach, based on using indirect methods to change the distribution of examples to reflect the costs of misclassification.

(ii) Cost-sensitive Bayesian networks via an amending approach, which involves amending the minimum description length measure used in constructing a network.

(iii) Cost-sensitive Bayesian networks via a Genetic algorithm, based on the use of genetic algorithms to construct a Bayesian Network that aims to minimise costs of misclassification.

The primary hypothesis of the present research stated that it would be possible to develop algorithms to learn cost-sensitive Bayesian networks, which on average are more cost-effective than current algorithms; including cost-sensitive decision learning tree algorithms such as: *MetaCost+J48*; cost-sensitive Bayesian networks which are available in WEKA (i.e. *MetaCost+BN*); or existing cost-insensitive Bayesian network learning algorithms (*TAN*). To evaluate this hypothesis, the research aimed to develop methods that analyse Bayesian networks that take account of misclassification costs and then utilise the empirical methods to assess the extent to which the hypothesis is true.

In this final chapter, a summary of how the research objectives have been addressed is provided in Section 6.1 with details of the achievements and main contributions from this research. Additionally, the limitations of the developed algorithms and directions for future work are presented in Section 6.2.

6.1 The research objectives revisited

This section presents the research objectives, as well as reviewing the extent to which they have been achieved and contributions made. The specific research objectives that were written in Chapter 1 are as follows:

- **To review the background of Bayesian networks learning algorithms, and analyse the types of this algorithm:** a survey of the foundation of Bayesian networks algorithms and the basic laws of probability were described in Chapter 2, Moreover, it presented how to learn the structure of Bayesian networks and their parameters.

- **To review the literature on cost-sensitive learning, analyse the most significant issues in current cost-sensitive learning algorithms, and identify the strategies and methods that used:** a survey of approaches to cost-sensitive learning was presented in Chapter 3, as a comprehensive literature review of the most appropriate methods that could be employed for developing cost-sensitive algorithm. Cost-sensitive algorithms were divided into three categories: direct methods, indirect methods, or optimisation methods. Most of the research to date has focused on developing cost-sensitive decision trees, while only three recent studies have developed algorithms for learning cost-sensitive Bayesian networks as described in Chapter 3. Therefore, the current study has

aimed to explore the development of algorithms for learning Bayesian networks for cost-sensitive classification.

- **To develop new cost-sensitive Bayesian network learning algorithms that aim to overcome the identified issues, which are based on direct, indirect, and optimisation methods:** new cost-sensitive Bayesian network algorithms have been developed and described in Chapter 4, which are:

    i. Cost-sensitive Bayesian network algorithm based on the indirect method via sampling approach.
    ii. Cost-sensitive Bayesian network algorithm based on the direct method via an amending approach.
    iii. Cost-sensitive Bayesian network algorithm based on the optimisation method via Genetic algorithms.

All of the algorithms mentioned above aim to minimise the misclassification cost, whilst maintaining accuracy.

- **To evaluate the new algorithms against existing cost-sensitive algorithms and measure performance, and compare the algorithms in terms of accuracy, and cost minimisation:** Chapter 5 presents the results of an empirical evaluation. This was undertaken in order to examine these algorithms and compare their performance with existing cost-sensitive algorithms, such as *MetaCost+j48*, and *MetaCost+BN*, and cost-insensitive Bayesian networks algorithms (*Tree Augmented Naive Bayes*). These alternatives algorithms have been implemented using Java based in the *WEKA* open source system. The algorithms have been evaluated through the use of 36 benchmark datasets, which have been studied previously by several researchers through different methods that have come from different domains. The evaluation was carried out using 16 cost ratios for two class labels [1:1,1:2,1:3,1:4, 2:1,2:2.., 4:4]. The experimental methodology used involved carrying out 10 random trials, and in each trial, the data was divided into 75% training and 25% testing. Through this, the results reported the averages together with the standard errors.

Furthermore, the results of an empirical comparison have been analysed in Chapter 5. Overall, the summary of the findings from these results show that:

(i) The three cost-sensitive Bayesian network algorithms outperform cost-insensitive Bayesian network algorithms in terms of cost in all datasets.

(ii) Algorithms, CS-BN via sampling, and CS-BN via amending have the same pattern as cost-sensitive algorithms *MetaCost+BN*. Indeed, CS-BN via sampling, and CS-BN via amending minimise cost and maintain the accuracy, as the number of misclassified rare class FN is less than the number of misclassified frequent class FP in most cases.

(iii) CS-BN via GA has a different pattern from other cost-sensitive algorithms, where it has been demonstrated this algorithm minimises cost and maintains good accuracy, because the number of misclassified rare class FN is similar to the number of misclassified frequent class FP in most of cases.

(iv) CS-BN via sampling works well when the original data distribution is very skewed. Where the data is biased to one class, this algorithm can function very well following the changes in the data distribution to reflect the misclassification costs, and increase the number of rare class instances. Thus, it is then possible to make these instances (rare instances) more important for classification than other instances. Comparing with *MetaCost+j48,* the performance of CS-BN via the sampling approach performs well in 29 out of the 36 datasets in terms of cost, and it performs well compared to the use of *MetaCost+BN* in all datasets in terms of cost. Through comparison of the current three algorithms, the performance of CS-BN via the sampling approach performs well in only 8 out of the 36 datasets in terms of cost.

(v) CS-BN via the amending approach works better than existing cost-sensitive algorithms as *MetaCost+BN* in all datasets, while comparing with *MetaCost+j48* it performs well in 28 out of 36 datasets. In particular, by comparing the current three algorithms, the performance of CS-BN via amending approach performs well in 13 out of the 36 datasets in terms of cost.

(vi) Comparing with *MetaCost+j48*, CS-BN via Genetic algorithms gives good results in 31 out of 36 datasets. While, CS-BN via GAs is better than our other algorithms in 24 out of the 36 datasets in terms of both costs and accuracy, and also performs better than *MetaCost+BN* for all the datasets in terms of cost and accuracy.

(vii) The accuracy of CS-BN via sampling, and CS-BN via amending is similar, although slightly less than the original accuracy based version of TAN, while CS-BN via GAs creates better results than all the other algorithms in terms of accuracy and cost as well.

6.2 Limitations and future work

The research has developed new cost-sensitive Bayesian network algorithms that aim to minimise the misclassification costs and which have been evaluated on 36 datasets that have a binary class label. The work presented has some limitations, which can be the subject of future works, including:

1. **Dealing with two classes:** all the experiments are performed on two class data; where the data that are used typically did not have more than two class label.

2. **Using misclassification cost:** all the developed algorithms are aimed to minimizing cost of misclassifications, but not test costs, which described on Section 3.2.

3. **Time consuming in CS-BN via GAs for high dimensional data:** if the data has lots of features, then the experiment 3 (CS-BN via GAs) will spend lots of time to find the optimal tree thus the training time will be long.

Therefore, there are several aspects that could be developed in the future, which include the following:

1. **Generality of the algorithms**: Although the developed algorithms were applied successfully for several datasets that have two class labels, there are certain limitations of these algorithms where they cannot be applied into datasets that have more than two class labels. Therefore, in additional future work, these algorithms could be developed to work on more than two class labels.

2. **Dealing with test costs**: Although the algorithms from the current study have aimed to minimise misclassification costs, they do not consider test costs. In particular, these algorithms can be extended in the future to include test costs when learning the structure of Bayesian networks.

In conclusion, the main contribution of this study is that three new algorithms for learning cost-sensitive Bayesian networks have been developed and evaluated. The evaluation of the algorithms shows advances in terms of minimising cost, with the CS-BN via GA performing the best. The comparison of the different methods, both existing and the new ones developed in this thesis, advance our knowledge of their relative merits.

References

Abe, N., Zadrozny, B., and Langford, J. (2004). An iterative method for multi-class cost-sensitive learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. pp. 3-11.

Aggarwal, C. C. (2014). *Data Classification: Algorithms and Applications*. CRC Press.

Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine learning*, *6*(1), pp.37-66.

Akaike, H. (1974). A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, *19*(6), pp. 716-723.

Asuncion, A., and Newman, D. (2007). UCI machine learning repository. Available at: http://archive.ics.uci.edu/ml/ (last accessed 25 October 2015).

Ben-Gal, I. (2007). Bayesian networks. *Encyclopedia of statistics in quality and reliability*. Wiley Online Library. Available at: http://onlinelibrary.wiley.com/doi/10.1002/9780470061572.eqr089/full (last accessed 25 October 2015).

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Bolstad, W. M. (2013). *Introduction to Bayesian statistics*. John Wiley & Sons.

Booker, L. B., and Hota, N. (2013). Probabilistic reasoning about ship images. In *the Second Annual Conference on Uncertainty in Artificial Intelligence*, University of Pennsylvania, Philadelphia, PA. *arXiv preprint arXiv:1304.3078*.

Bouckaert, R. R. (2004). *Bayesian network classifiers in weka*. Department of Computer Science, University of Waikato.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Buntine, W. (1991). Theory refinement on Bayesian networks. In *Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc. pp. 52-60.

Campos, L. M. (2006). A scoring function for learning bayesian networks based on mutual information and conditional independence tests. *The Journal of Machine Learning Research*, *7*, pp. 2149-2187.

Carvalho, A. M. (2009). Scoring functions for learning Bayesian networks. *Inesc-id Tec. Rep.*

Charniak, E., and Goldman, R. P. (1989). A Semantics for Probabilistic Quantifier-Free First-Order Languages, with Particular Application to Story Understanding. In *the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, Michigan, USA. In *IJCAI* ,Vol. 89, pp. 1074-1079.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, pp.321-357.

Cheng, J., and Greiner, R. (1999). Comparing Bayesian Network Classifiers. UAI 1999, pp.101-108.

Cheng, J., and Greiner, R. (2001). Learning bayesian belief network classifiers: Algorithms and system. In *Advances in Artificial Intelligence.* Springer Berlin Heidelberg. pp. 141-151.

Cheng, J., Bell, D.A. and Liu, W. (1997). An algorithm for Bayesian belief network construction from data. In Proceedings of AI & STAT'97, Florida, pp.83-90.

Chickering, D.M .(2002).Optimal Structure Identification with Greedy Search. *Journal of Machine Learning Research 3* , pp. 507-554.

Chow, C. K., and Liu, C.N. (1968). Approximating discrete probability distributions with dependence trees, IEEE Transactions on Information Theory IT-14 (3), pp, 462–467.

Cios, K. J., and Moore, G. W. (2002). Uniqueness of medical data mining. *Artificial intelligence in medicine*, *26*(1), pp. 1-24.

Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the twelfth international conference on machine learning* . pp. 115-123.

Cooper, G.F. and Herskovits, E. (1992). A Bayesian Method for the induction of probabilistic networks from data. *Machine Learning*, pp. 309-347.

Corani, G., Antonucci, A., and Zaffalon, M. (2012). Bayesian networks with imprecise probabilities: theory and application to classification. In *Holmes, D. E., Jain, L. C. (Eds), Data Mining: Foundations and Intelligent Paradigms* (Volume 1: Clustering, Association and Classification), Springer-Verlag, Berlin, pp. 49–93.

Cormen , T. H. Leiserson, C. E. and Rivest. R. L. (1990). Introduction to Algorithms. MIT Press, Cambridge, MA.

Cover, T. M., and Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

Dash, M., and Liu, H. (1997). Feature selection for classification. Intelligent data analysis, 1(1), 131-156.

Dasgupta, S. (1999).Learning polytrees. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence* , pp. 134-141.

Davis,Jj. V., Jungwoo, H. and 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*, pp.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, pp. 155–164.

Drummond, C., and Holte, R. C. (2003). C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II* (Vol. 11).

Duda, R.O., and Hart.P.E.(1973). *Pattern classification and scene analysis*. (Vol. 3). New York: Wiley.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*. LAWRENCE ERLBAUM ASSOCIATES LTD. Vol. 17, No. 1, pp. 973-978.

Estabrooks, A. Jo, T. and Japkowicz, N. (2004). A Multiple Resampling Method for Learning from Imbalanced Data Sets. *Computational Intelligence*, vol. 20, pp. 18-36.

Ezawa, K. J., and Schuermann, T. (2015). Practice of Bayesian Networks Data Mining Lab 4. Available at : http://www.engr.uvic.ca/~seng474/lab4.ppt . (last accessed 25 October 2015).

Fan, W., Chu, F., Wang, H., and Yu, P. S. (2002). Pruning and dynamic scheduling of cost-sensitive ensembles. In *Proceedings of the National Conference on Artificial Intelligence*. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press;. pp. 146-151.

Fan, W., Stolfo, S. J., Zhang, J., and Chan, P. K. (1999). AdaCost: misclassification cost-sensitive boosting. In *ICML* . pp. 97-105.

Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, pp. 309-368.

Fisher, R. A. (1997). On an absolute criterion for fitting frequency curves. *Statistical Science*, pp. 39-41.

Freitas, A., Costa-Pereira, A., and Brazdil, P. (2007). Cost-sensitive decision trees applied to

medical data. In *Data Warehousing and Knowledge Discovery* (pp. 303-312). Springer Berlin Heidelberg.

Freund, Y., and Schapire, R.E. (1996). Experiments with a new boosting algorithm. *13th International Machine Learning workshop then conference*, Bari, Italy, pp. 148-156.
Friedman, N., Geiger ,D., and Goldszmidt, M. (1997). Bayesian Network Classifiers. *Machine Learning*, 29(2-3), pp. 131–163.

Friedman, N., & Goldszmidt, M. (1998). Learning Bayesian networks with local structure. In *Learning in graphical models*. Springer Netherlands. pp. 421-459.

Funahashi, K. I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural networks*, *2*(3), pp.183-192.

Gao, Y., Wang, Z., and Chen, Y. (2008). Cost-Sensitive Parameters Estimation Method of Bayesian Networks. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on* IEEE. pp. 1-4.

Green, M. T. (2010). A comparison of methods for learning cost-sensitive classifiers.

Gurland, J., and Tripathi, R. C. (1971). A simple approximation for unbiased estimation of the standard deviation. *The American Statistician*, *25*(4), pp. 30-32.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, *11*(1), pp.10-18.

Han, J., Kamber, M., and Pei, J. (2015). Data Mining: Concepts and Techniques. *University of Illinois at Urbana-Champaign & Simon Fraser University*. (last accessed 25 October 2015)

Hansson, O., and Mayer, A. (1989). Decision-theoretic control of search in BPS. In *Proceedings of the AAAI Symposium on AI and Limited Rationality*.

He, H., and Garcia, E. (2009). Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, *21*(9), pp. 1263-1284.

Heckerman, D. (1997). Bayesian networks for data mining. *Data mining and knowledge discovery*, *1*(1), pp. 79-119.

Heckerman, D. (2008). A Tutorial on Learning with Bayesian Networks. Jonathan Paranada, Probability and Statistics. pp. 33-82.

Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine learning*, *20*(3),pp. 197-243.

Heckerman, D., Mamdani, A., and Wellman, M. P. (1995). Real-World Applications of

Bayesian Networks. *Communications of the ACM*, 38(3), pp. 24-68.

Hong, S. A.(2007). Structure Learning in Bayesian Networks (mostly Chow-Liu). Available athttp://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&frm=1&source=web&cd=3&ved=0CCgQFjACahUKEwjJhLf5xOrIAhWDOBQKHcJ0ACs&url=http%3A%2F%2Fwww.cs.cmu.edu%2F~guestrin%2FClass%2F15781%2Frecitations%2Fr10%2F11152007chowliu.pdf&usg=AFQjCNHQq6-qLRPTjYHzII0JleiwkWVsPQ . (last accessed 29 October 2015).

Hui, L. (2015). Class Imbalance vs. Cost-Sensitive Learning. University of Ottawa. http://webcache.googleusercontent.com/search?q=cache:yESJnS6VXlEJ:https://www.site.uottawa.ca/~nat/Courses/csi5388/Presentations/ClassImbalanceVs.CostSensitiveLearning.ppt+&cd=1&hl=en&ct=clnk&gl=uk .(last accessed 28 Oct 2015).

Jiang, L., Li, C., and Wang, S. (2014). Cost-sensitive Bayesian network classifiers. *Pattern Recognition Letters*, *45*, pp. 211-216.

Kantardzic, M. (2011). *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons.

Kearns, M. (1988). Thoughts on hypothesis boosting. *Unpublished manuscript*, *45*, 105.

Kenett, R. S. (2012). Applications of Bayesian networks. *Available at SSRN 2172713*.

Kjaerulff, U. B., and Madsen, A. L. (2008). Bayesian networks and influence diagrams. *Springer Science+ Business Media*, *200*, 114.

Kong, G., Xia, Y., and Qiu, C. (2014). Cost-Sensitive Bayesian Network Classifiers and Their Applications in Rock Burst Prediction. In *Intelligent Computing Theory* . Springer International Publishing. pp. 101-112.

Kothari, C. R. (2011). *Research methodology: Methods and techniques*. New Age International.

Kountz, B., Miryala, A., Scarlett, K., and Zell, Z.(2011). Bayes Rule, conditional probability, independence. Available at: https://controls.engin.umich.edu/wiki/index.php/File:Contengency_table.jpg. (last accessed 25 October 2015).

Kohavi, R., and John, G. H. (1997). Wrappers for feature subset selection. Artificial intelligence, 97(1), 273-324.

Kubat, M. and Matwin, S. (1997). Addressing the Curse of Imbalanced Training Sets: One Sided Selection. In Proceedings of the Fourteenth International Conference on Machine Learning, pages 179-186, Nashville, Tennesse. Morgan Kaufmann.

Langley, P., Iba, W., and Thompson, K. (1992). An analysis of Bayesian classifiers. In *AAAI* (Vol. 90), pp. 223-228.

Laskey K. B. (2015). Bayes Nets. Available at http://www.bayesnets.com/ .(last accessed 27 October 2015).

Laurikkala, J. (2001). *Improving identification of difficult small classes by balancing class distribution*. Springer Berlin Heidelberg, pp. 63-66.

Lauritzen, S. L., and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 157-224.

Ling, C. X., and Sheng, V. S. (2010). Cost-sensitive learning. In *Encyclopedia of Machine Learning* .Springer US. pp. 231-235.

Ling, C. X., Sheng, V. S., and Yang, Q. (2006). Test strategies for cost-sensitive decision trees. *Knowledge and Data Engineering, IEEE Transactions on*, *18*(8), pp.1055-1067.

Ling, C. X., Yang, Q., Wang, J., and Zhang, S. (2004). Decision trees with minimal costs. In *Proceedings of the twenty-first international conference on Machine learning* .ACM. p. 69.

Ling, C., Sheng, V., and Yang, Q. (2006). Test strategies for cost-sensitive decision trees. IEEE Transactions on Knowledge and Data Engineering, 18(8), pp. 1055-1067.
Liu, X. (2007). A New Cost-Sensitive Decision Tree with Missing Values. Asian Journal of Information Technology, 6(11), pp.1083–1090.

Lomax, S., and Vadera, S. (2013). A survey of cost-sensitive decision tree induction algorithms. *ACM Computing Surveys (CSUR)*, *45*(2), 16.

Maes, S., Tuyls, K., Vanschoenwinkel, B., and Manderick, B. (2002). Credit card fraud detection using Bayesian and neural networks. In *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*.

Maimon, O., and Rokach, L. (2005). *Data mining and knowledge discovery handbook.* (Vol.2). New York: Springer.

Maloof, M. A. (2003). Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 workshop on learning from imbalanced data sets II*. Vol. 2, pp. 2-1.

Margineantu, D. (2000). On class probability es- timates and cost-sensitive evaluation of classifiers. *In Workshop Notes, Workshop on Cost-Sensitive Learning, International Conference on Machine Learning*.

Margineantu, D., and 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 (last accessed 25 October 2015).

Meek, C. (2015).An Overview of Learning Bayes Nets From Data. Microsoft Research. Available at http://research.microsoft.com/en-us/people/meek/ .(last accessed 26 October 2015).

Michie, D., Spiegelhalter, D. J., and Taylor, C. C. (1994). Machine learning, neural and statistical classification.

Mitchell, T. M. (1997). Does machine learning really work?. *AI magazine*, *18*(3), 11.

Morris, R. D. (2003). Bayesian Research at the NASA Ames Research Center, Computational Sciences Division.

Nashnush, E., and Vadera, S. (2014). Cost-Sensitive Bayesian Network Learning Using Sampling. In *Recent Advances on Soft Computing and Data Mining* .Springer International Publishing. pp. 467-476.

Neapolitan, R. E. (2004). *Learning bayesian networks* (Vol. 38). Upper Saddle River: Prentice Hall.

Norsys software Coper. (2015). Available at https://www.norsys.com/netica.html . (last accessed 25 October 2015).

Norton, S. W. (1989). Generating Better Decision Trees. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. IJCAI-89, 20-26 August 1989, Detroit,Michigan, USA, pp. 800-805.

Núñez, M. (1991). The use of background knowledge in decision tree induction. *Machine learning*, *6*(3), pp. 231-250.

Omielan, A., and Vadera, S. (2012). ECCO: A New Evolutionary Classifier with Cost Optimisation. School of computing ,Scince and Engininring, University of Salford , Salford M5 4WT,UK , pp.1.

P. Dagum and M. Luby.(1993). Approximating probabilistic inference in Bayesian belief networks is NPhard. *Artificial Intelligence*, vol. 60, pp. 141-153.

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., and Brunk, C. (1994). Reducing misclassification costs. In *Proceedings of the 11th International Conference on Machine Learning*. New Brunswick, New Jersey, USA, pp. 217–225.

Pearl, J. (1988). Embracing causality in formal reasoning . *In AAAI*, Artificial Intelligence, 35, pp. 259-271.

Pearl, J. (1988; 2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Pearl, J. (1993). Belief networks revisited. *Artificial intelligence*, *59*(1), pp.49-56.

Pearl, J. (2001). Bayesianism and causality, or, why I am only a half-Bayesian. In *Foundations of bayesianism* . Springer Netherlands, pp. 19-36.

Phua, C., Lee, V., Smith, K., and Gayler, R. (2010). A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*.

Pourret, O., Naim, P., and Marcot, B. (2008). *Bayesian Networks: A practical Guide to applications* (Vol. 73). John Wiley & Sons.

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

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), pp. 81-106.

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, pp.168-201.

Rajasekar, S., Philominathan, P., and Chinnathambi, V. (2006). Research methodology. *arXiv preprint physics/0601009*. *).* Available at: https://www.scribd.com/deleted/6949151 .(last accessed 25 October 2015)

Ramos, A. A. (2006). The minimum description length principle and model selection in spectropolarimetry. *The Astrophysical Journal*, *646*(2), 1445.

Rayner , J.C. W. and Best, D.J.(1989). Smooth Tests of Goodness of Fit. Oxford University Press, Inc., ISBN 0-19-505610-8.

Rish, I.(2015).Advances in Bayesian Learning, Learning and Inference in Bayesian Networks. IBM T.J. Watson Research Center.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, *14*(5), pp. 465-471.

Robinson, R.W.(1973). Counting labeled acyclic digraphs, in *F.Harary (Ed.*), New Directions in the Theory of Graphs, Academic Press, New York, pp. 239-273.

Robinson, R.W.(1977). Counting unlabeled acyclic digraphs, in *C.H.C. Little (Ed.),* Combinatorial Mathematics V, volume 622 of Lecture Notes in Mathematics, Springer-Verlag, Australia, pp. 28-43.

Rygielski, C., Wang, J. C., and Yen, D. C. (2002). Data mining techniques for customer relationship management. *Technology in society*, *24*(4), pp. 483-502.

Salama, K. M., and Freitas, A. A. (2013). Learning Bayesian network classifiers using ant colony optimization. *Swarm Intelligence*, *7*(2-3), pp. 229-254.

Santos-Rodríguez, R., García-García, D., and Cid-Sueiro, J.(2009). Cost-sensitive classification based on Bregman divergences for medical diagnosis. In *Machine Learning and Applications, ICML*, pp. 551-556.

Sawaal. (2015). Available at: http://www.sawaal.com/probability-questions-and-answers/a-bag-contains-2-red-3-green-and-2-blue-balls-two-balls-are-drawn-at-random-what-is-the-probability-_3271 . (last accessed 25 October 2015).

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,pp. 1401- 1406.

Schapire, R. E. and Singer, Y. (1999). Improved Boosting Algorithms Using Confidence-Rated Predictions. Machine Learning 37, (3), pp. 297-336.

Schum, D. A. (2001). *The evidential foundations of probabilistic reasoning*. Northwestern University Press.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, *6*(2), pp. 461-464.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, *6*(2), pp.461-464.

Sebe, N. ,Cohen, I., and Garg, T.S. (2005). Machine Learning in Computer Visison. *Huang, Springer Verlag, ISBN* 1-4020-327, pp.4-9.

Shannon, C.E. and Weaver, W. (1949). The mathematical theory of communication. University of Illinois Press, Urbana, Illinois.

Sheng, V. S., and Ling, C. X. (2006). Thresholding for making classifiers cost-sensitive. In *Proceedings of the national conference on artificial intelligence.* (Vol. 21, No. 1, p. 476). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Spiegelhalter, D. J., Franklin, R., and Bull, K. (1989). Assessment, Criticism, and Improvement of Imprecise Probabilities for a Medical Expert System. In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 285-294.

Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, prediction, and search* (Vol. 81). MIT press.

Statistical Visual Computing Lab. UCSD. (2015). http://www.svcl.ucsd.edu/projects/sop_boost/ . (Last accessed 25 October 2015).
Suna, Y., Kamela,S. M., Andrew K.C.Wongb, K., Wang, Y.(2006). Cost-sensitive boosting for classification of imbalanced data, Electrical and Computer Engineering Department, University of Waterloo, Waterloo, Ontario, Canada.

Tan , M. and 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, pp. 392-395.

Ting, K. and Zheng, Z.(1998a). Boosting cost-sensitive trees. Lecture Notes in Computer Science, 1532, pp. 244–255.

Ting, K. and Zheng, Z.(1998b). Boosting Trees for Cost-Sensitive Classifications. In Machine Learning: ECML-98 10th European Conference on Machine Learning, Chemnitz, Germany. Springer, pp. 190-195.
Ting, K. M. (2002). An instance-weighting method to induce cost-sensitive trees. *Knowledge and Data Engineering, IEEE Transactions on*, *14*(3),pp. 659-665.

Turney, P. D. (1995). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of artificial intelligence research*, pp. 369-409.

Turney, P.(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, pp. 15-21.

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

Vandel, J., Mangin, B., and Givry, S. (2012). New local move operators for Bayesian network structure learning. *Proceedings of PGM-12, Granada, Spain*.

Weiss, G. M., and Provost, F. (2001). The effect of class distribution on classifier learning: an empirical study. *Rutgers Univ*.

Weiss, G. M., McCarthy, K., and Zabar, B. (2007). Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs?. In *DMIN* , pp. 35-41.

WIKIbooks. (2015). Open books for an open word Probability/Combinatorics. Available at https://en.wikibooks.org/wiki/Probability/Combinatorics .(last accessed 31 October 2015)

Witten, I. H., and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Wolpert, DH. (1996).The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation*. pp.1341-1390

Zadrozny, B., and Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* , ACM, pp. 204-213.

Zadrozny, B., Langford, J., and Abe, N. (2003b). Cost-sensitive learning by cost-proportionate example weighting. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on* IEEE. pp. 435-442.

Zadrozny, B., Langford, J., and Abe, N. (2003a). A simple method for cost-sensitive learning.

Zhang, Y., Meratnia, N., and Havinga, P. (2010). Outlier detection techniques for wireless sensor networks: A survey. *Communications Surveys & Tutorials, IEEE*, *12*(2), pp.159-170.

Zibran, M. F. (2007). Chi-Squared test of independence. *Department of Computer Science, University of Calgary, Alberta, Canada.*

Zweig, G., and Russell, S. (1998). Speech recognition with dynamic Bayesian networks. AAAI, p.173.

Appendix A

A1: Connections in a BN structure

The *inference process* in Bayesian networks is based on *d-separation* concepts (Preal, 1993); where two sets of nodes are conditionally independent given a set of evidence. The structure of the graph represents the conditional independence relations, as Pearl in (1988) stated:

"*Each variable is conditionally independent of all its non-descendants given the value of all its parents*".

Where, the concept of *d-separation* determines the minimum amount of information needed to process a query during the exact inference in a Bayesian network, it decides which conditional independence relations are implied by a directed acyclic graph of the Bayesian network. For example, in Figure A1.1 when C is known, then A and B are conditional independents given C. When one says A and B are *d-separated* by a set of evidence, for C every undirected path from A to B is "blocked" (Pearl, 1988).

There are three types of connections between nodes, which allow the transfer of information through the nodes. These connections help to follow how a change of certainty in one node may change the certainty of other nodes (Kjaerulff and Madsen, 2008). Thus, there are 4 types of connections in any BNs' structure.

i. Causal chain (Serial connection)

Two nodes in a path connected as *Tail to Head*, where, in Figure A1.1, where both of A, and B are dependent if C is unknown; while, both of A, and B are conditional independent given C, when C is known.

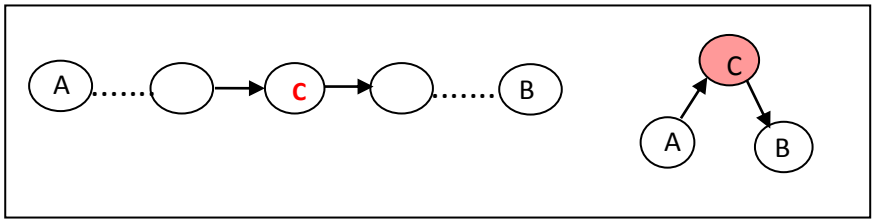


Figure A1.1: Blocking Tail- Head path (serial connection)

ii. Common causes (Diverging connection)

Two nodes in a path as "*Tail to Tail*", as illustrated in Figure A1.2, where both of A, and B are dependent if C is unknown, while, both of A, and B are conditional independent given C when Z is known.

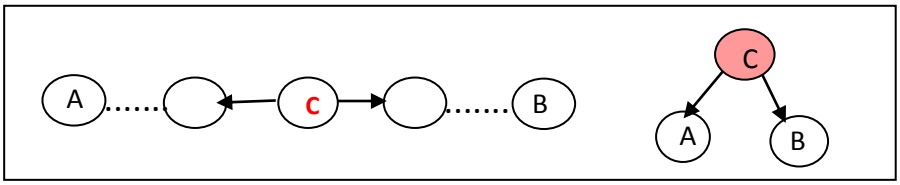


Figure A1.2: Blocking Tail - Tail path (Diverging connection)

iii. Common effects (Convergence connection)

Two nodes in a path are "*Head to Head*", as illustrated in Figure A1.3 where both of A, and B are conditional dependent if C or any of its descendants are known. While, both of A, and B are independent when C or any of its descendants are unknown (Kjaerulff and Madsen, 2008).

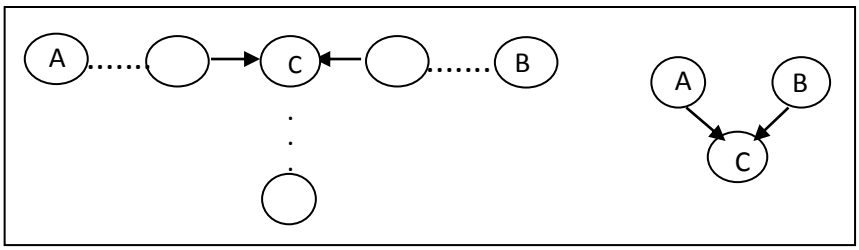


Figure A1.3: Opening Head- Head path (Convergence connection)

A2: Example to illustrate Propagation of information in the Alarm problem

This part describes how the information flows in a Bayesian network, by using the Alarm problem example (Pearl, 1988). Figure A2.1 presents the *alarm* example (Pearl 1988) and its representation in *Netica*, which is a software for Bayesian networks (Norsys, 2015). The alarm problem is stated as follows (Pearl 1988): Mr. Holmes is working in his office when he receives a phone call from his neighbour Dr. Watson who tells him that Holmes' burglar alarm has gone on; where a burglary or earthquake make the alarm goes on, John or Mary also call to report the alarm.

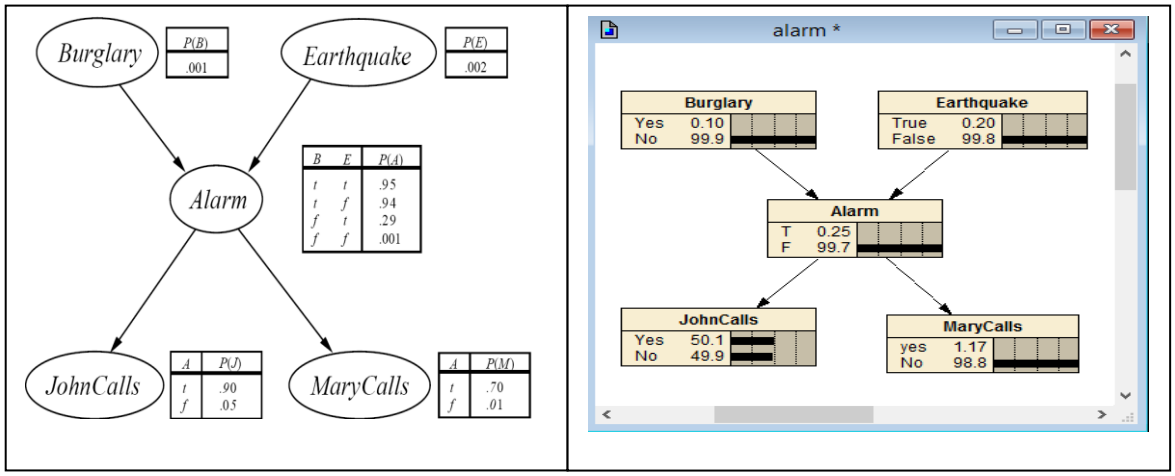
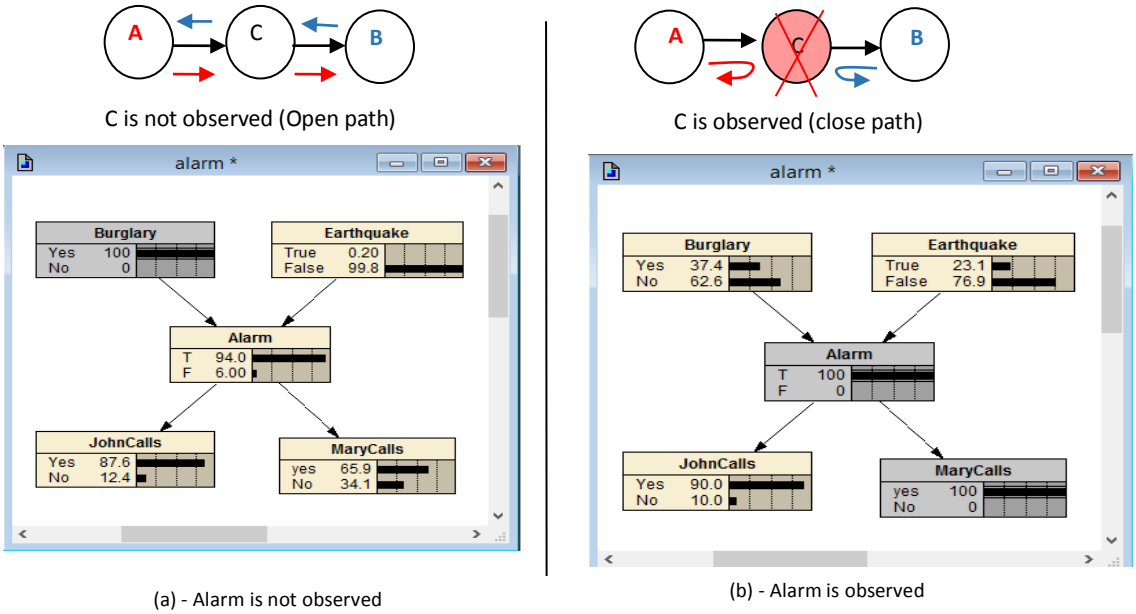


Figure A2.1: Alarm network (pearl, 1988).

From the graph above, the Burglary and Earthquake events are dependent on Alarm events, while both the Burglary and Earthquake events are *independent* of each other. In particular, there are three methods to connect the nodes in a BN and these connections represent the c*onditional connections* in a Bayesian Network. In particular, there are three types of connection that exist in this part between variables in the alarm problem:

i. Serial connection

If the Alarm is not evidence, then both calls (Burglary and Mary's) are dependent and knowing that Burglary=yes would increase the belief on Mary calls= yes, as shown in Figure A2.2(a). In contrast, if the Alarm node is observed, then that would increase the belief of both calls from the Burglary and Mary, but if knowing that burglary has taken place, then that would not change the belief on Mary's call because the path is closed, as shown in Figure A2.2(b).



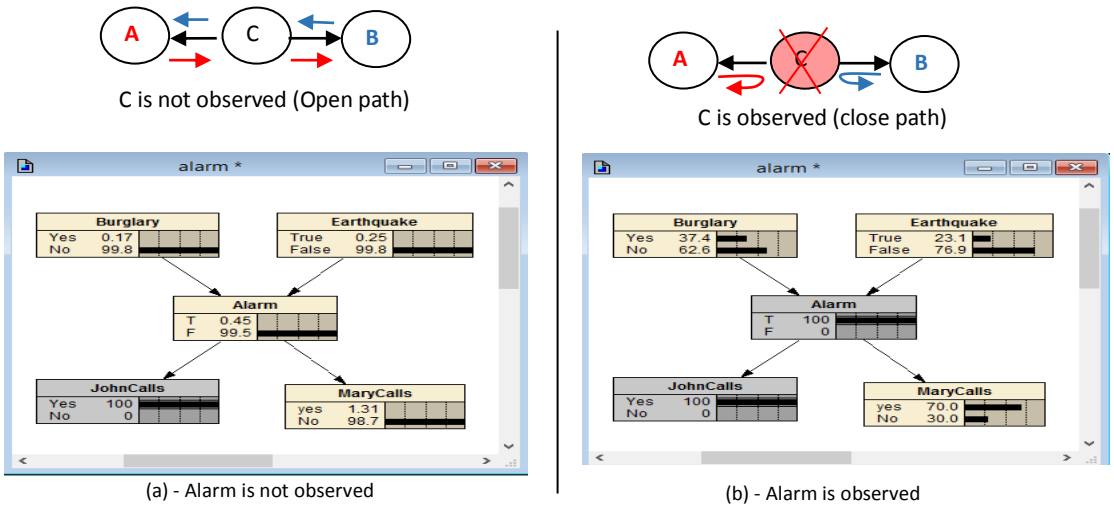
(a) - Alarm is not observed

(b) - Alarm is observed

Figure A2.2: Serial connection in alarm problem

ii. Diverging connection

If the Alarm is not evidence then both John and Mary called dependently and knowing one of them would increase the belief on the other one, as shown in Figure A2.3 (a). On the other hand, if the Alarm node is observed, that would change the belief of both John and Mary calling, although if there was further knowledge that John's call had taken place then that would not increase the belief on Mary's call, as shown if Figure A2.3 (b).



(a) - Alarm is not observed

(b) - Alarm is observed

Figure A2.3: Diverging connection

iii. Converging connection

As shown in Figure A2.4 (a), when the Alarm or its parent (Mary calls and John calls) is not observed then both of Burglar and Earthquake are independent. On the other hand, as shown in Figure A2.4 (b) when node C or any of its descendant as D is observed then A and B are conditionally dependent; where, observing C or its descendent as D opens the information path between A, and B. For example, if the Alarm node that is observed went on, then that would increase the belief of both a burglary and earthquake, and through further knowledge that there has been a burglary, then that would decrease the belief of an earthquake, because both the burglary and earthquake are dependent when the Alarm is observed, as show in Figure A2.4 (b).

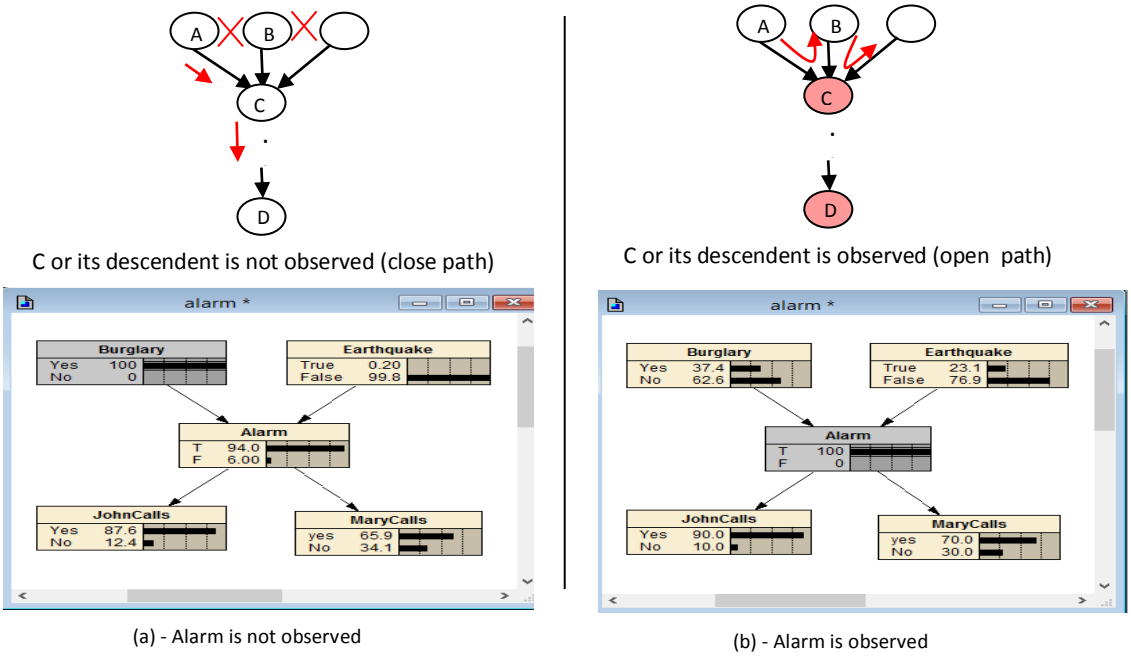


Figure A2.4: Diverging connection

Appendix B

B1: Example of learning a TAN using the play-tennis dataset

This appendix presents an example to illustrate Friedman et al.'s (1997) algorithm for learning TANs, which was presented in Chapter 2, Section 2.4.1.3.2. Table B1.1 presents the play-tennis training dataset that is used to illustrate the steps of the algorithm. There are two major parts: *learning the structure* and then *calculating the parameters,* which are illustrated on the data chart below.

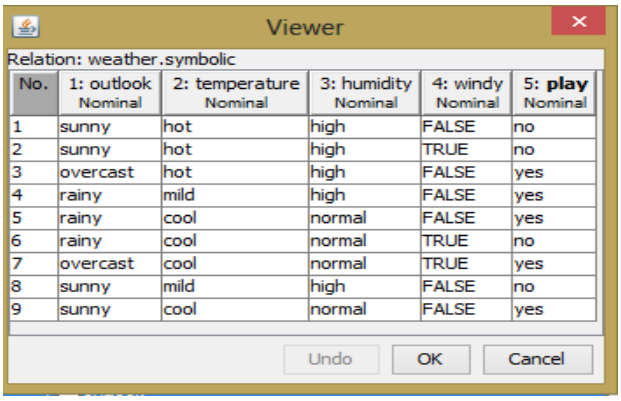


Table B1.1: Play-tennis training datasets

i. Learn TAN Structure

Learning the TAN structure involves the following steps:

Step 1: Compute Conditional Information

The first step calculates the information between two nodes based on the class node by using the MDL score that is presented in equations (2.15), and (2.16), which is based on the LL function that is given in equation (2.8). Moreover, the MDL score should be calculated between:

- MDL($X_i|C$): each node in the network and class node (dependent class).
- MDL($X_i|X_j, C$): each pair of nodes in the network, without class node (dependent nodes).

- The weight between each pair of nodes represents the difference between MDL dependent nodes, and MDL dependent class.

1) **Calculate the MDL dependent class**: This represents the score between a selected node and class node. For example, if the selected node is 'outlook', and the class node is 'play', then to calculate MDL(outlook, play) one needs to calculate LL function initially as presented in equation (2.8):

$$\text{LL(outlook| play)} = \sum_{i=1}^{3}\sum_{k=1}^{2} \text{P(outlook, play)}\ log\ (\frac{P(\text{outlook, play})}{P(\text{ play})})$$

Where, i represents the selected node values, the 'outlook' has (sunny, overcast, and rainy), and k represents the number of attributed values of the class node, which are (yes, no). From the data in Table B1.1 the results will be:

$$\text{LL(outlook| play)} = -7.524$$

Then, calculate MDL for the dependent class function, as given in equation (2.11):

$$\text{MDL(outlook| play)} = LL(\text{outlook, play}) - \frac{log\ N}{2}\ |B|$$

Where, N is the number of all instances in the data, it comes to 9 in this example. |B| is the number of parameters = number of selected node values * number of class values.

$$\text{MDL(outlook|play)} = -7.524 - \frac{log\ (9)}{2} * |3 * 2| = -11.918$$

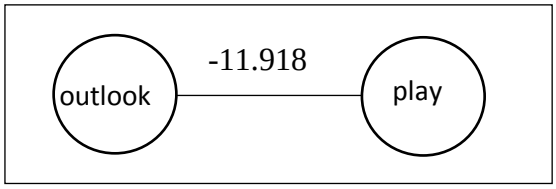


Figure B1.1: Score between 'outlook' and class node

2) **Calculate MDL dependent nodes:** This represents the score between a selected node and another node based on the class node. For example, when the selected node is 'outlook', and the another node is 'windy' based on the class label 'play'. Then, to calculate MDL(outlook|windy, paly), one needs to calculate the LL function initially, as given in equation (2.8):

$$\mathrm{LL}(\text{outlook}|\text{wind}, \text{play})$$

$$= \sum_{i=1}^{3}\sum_{j=1}^{2}\sum_{k=1}^{2} \mathrm{P}(\text{outlook}, \text{windy}, \text{play})\ log\ (\frac{P(\text{outlook}, \text{windy}, \text{play})}{P(\text{windy}, \text{play})})$$

Where, i represents the selected node values, the 'outlook' has (sunny, overcast, and rainy), j represents the attributes values of another node 'wind', which are (false, and true), and k represents the number of attributed values of the class node, which are (yes, no). From the data in Table B1.1 the results will be:

$$\mathrm{LL}(\text{outlook}|\text{windy}, \text{play}) = -5.5452$$

Then, calculate MDL for the dependent nodes, as given in equation (2.11):

$$\mathrm{MDL}(\text{outlook}|\text{windy}, \text{paly}) = LL(\text{outlook}, \text{windy}, \text{paly}) - \frac{log\ N}{2}\ |B|$$

Where, N is the number of all instances in data, 9 is the amount in this example. |B| is the number of parameters = number of selected node values $*$ number of cardinality values.

$$\mathrm{MDL}(\text{outlook}|\text{windy}, \text{play}) = -5.5452 - \frac{log\ (9)}{2} * |3 * 4| = -14.33$$

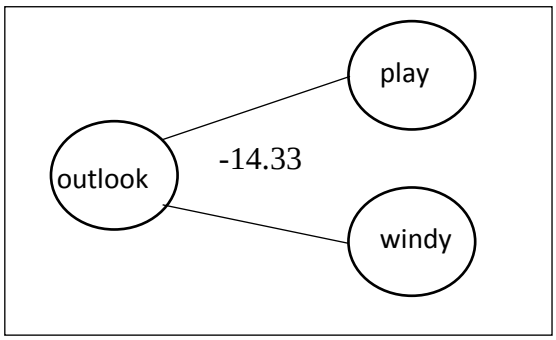


Figure B1.2: Score between 'outlook' and 'wind' based on class node

3) **Calculate the weight**: The weight or the dependency between two nodes represents as the difference between the dependent nodes' score, and dependent class score as demonstrated in:

Weight (outlook, windy)= $\mathrm{MDL}(\text{outlook}|\text{wind}, \text{paly}) - \mathrm{MDL}(\text{outlook}|\ \text{paly})$

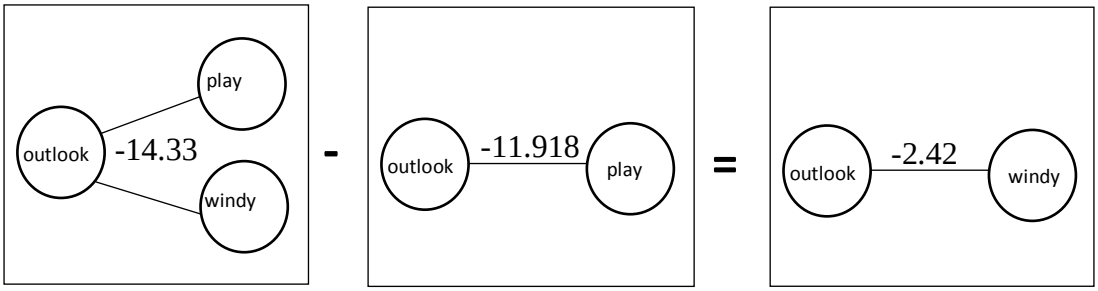


Figure B1.3: weight between ‘outlook’ and ‘wind’ node

Then, do the first step for all other nodes, as after that it will be possible to acquire the weights between all pairs, as shown in Figure B1.4. Also, the arcs between the nodes represent weight (dependency) between these nodes.

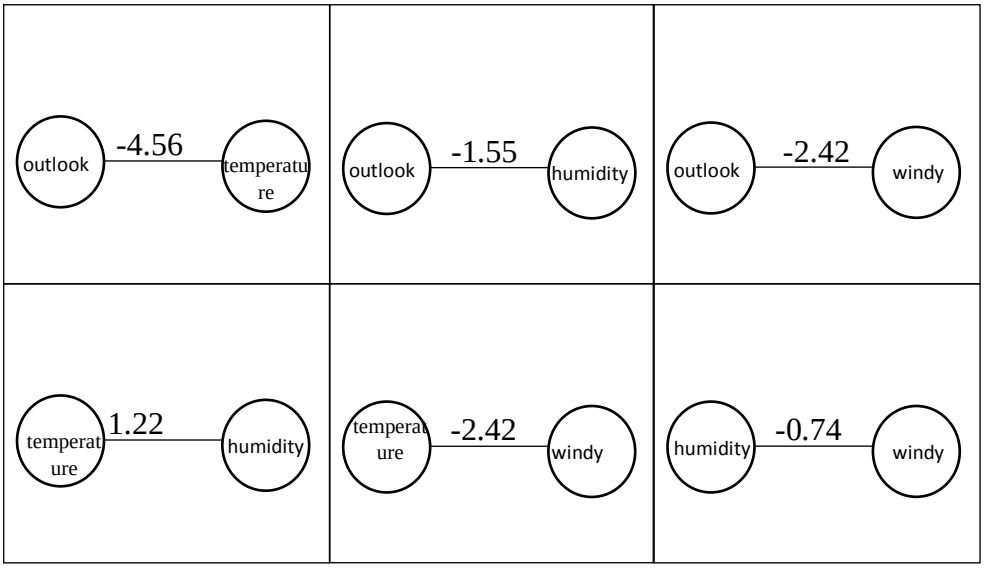


Figure B1.4: weight (dependency) between all pairs

Step2: Build a complete undirected graph: An undirected graph is constructed, where the edges are the dependency (weight) between the nodes, as shown in Figure B1.5.

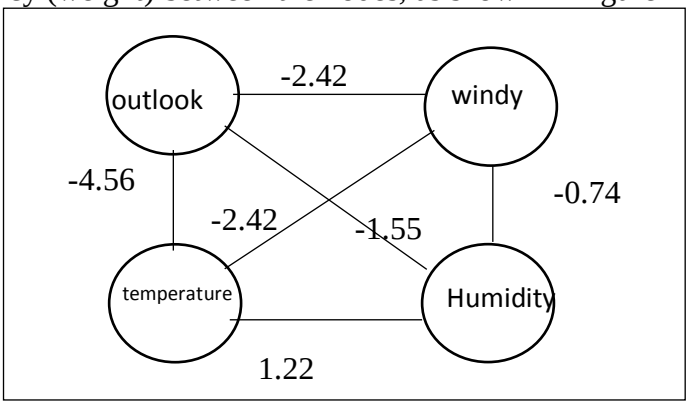


Figure B1.5: undirected graph

***Step3: Apply (MWST) algorithm**:* Find a maximal spanning tree between nodes by running a maximum-weight spanning tree (MWST) algorithm (Cormen et al., 1990), Figure B1.6 shows how the MWST finds the tree with the greatest total weight.

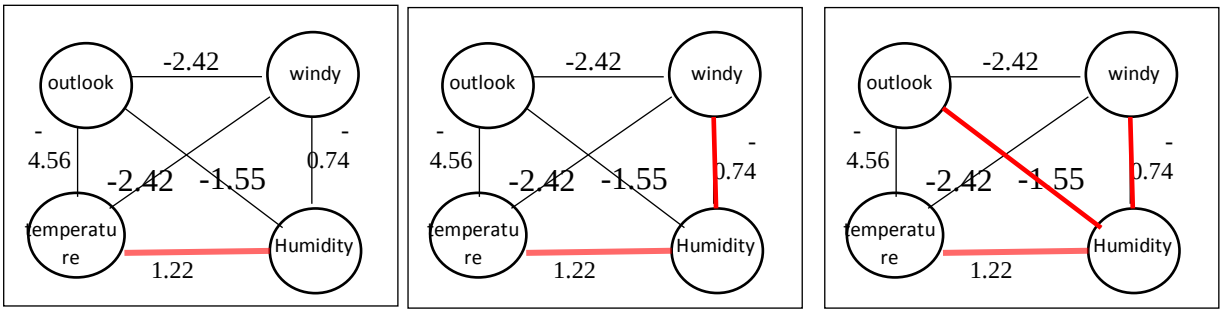


Figure B1.6: How MWST finds the tree with the greatest total weight in play- tennis dataset

***Step4: Convert to directed tree**:* The undirected graph is converted to a directed graph by choosing the root of the first maximum connection in the previous step, which is 'temperature', then adding a direction to the next connection if it does not lead to a cycle.

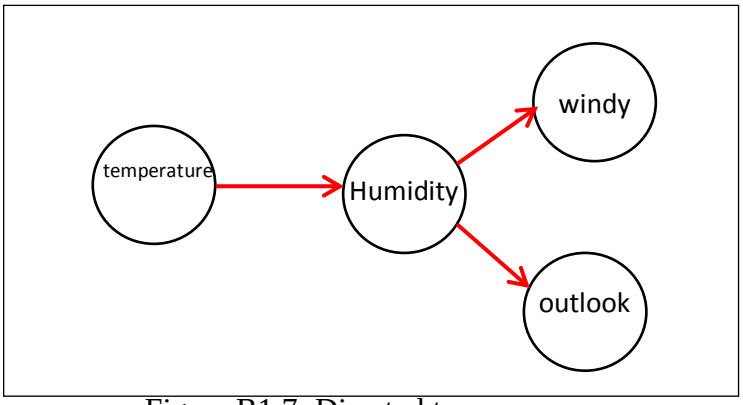


Figure B1.7: Directed tree

***Step5: Add the class label as root**:* The class label node 'play' is added as the parent (root) node for all attributes, to get the TAN structure, as shown in Figure B1.8.

ii. Learn parameters

Once the structure of a TAN has been learned, a s*imple estimator* is used for estimating the conditional probability tables of the TAN (Bouckaert, 2004). Obviously, the last step is to learn the parameters for each node with its parent by using equation (2.17):

$$\mathrm{P}(X_i|X_j) = \frac{N_{ij} + \alpha}{\mathrm{N} + (\alpha * \mathrm{n}_{X_i})}$$

Where, $\alpha = 0.5$ represents the initial count on each value to avoid 0, and n_{X_j} represents the number of attributes value of node X_i. For example, in Figure B1.8, in a CPT to learn the parameter between node 'temperate=hot' given 'play=yes', is calculated as:

$$\mathrm{P}(\text{'temperate} = \text{hot'} \mid \text{'play} = \text{yes'}) = \frac{\mathrm{P}(\text{'temperate} = \text{hot'}, \text{'play} = \text{yes'}) + 0.5}{\mathrm{P}(\text{play} = \text{yes}) + (0.5 * 3)}$$

$$\mathrm{P}(\text{'temperate} = \text{hot'} \mid \text{'play} = \text{yes'}) = \frac{1 + 0.5}{5 + (0.5 * 3)} = \frac{1.5}{6.5} = 0.231$$

Figure B1.8 illustrates learning structure and parameters of TAN on play-tennis dataset.

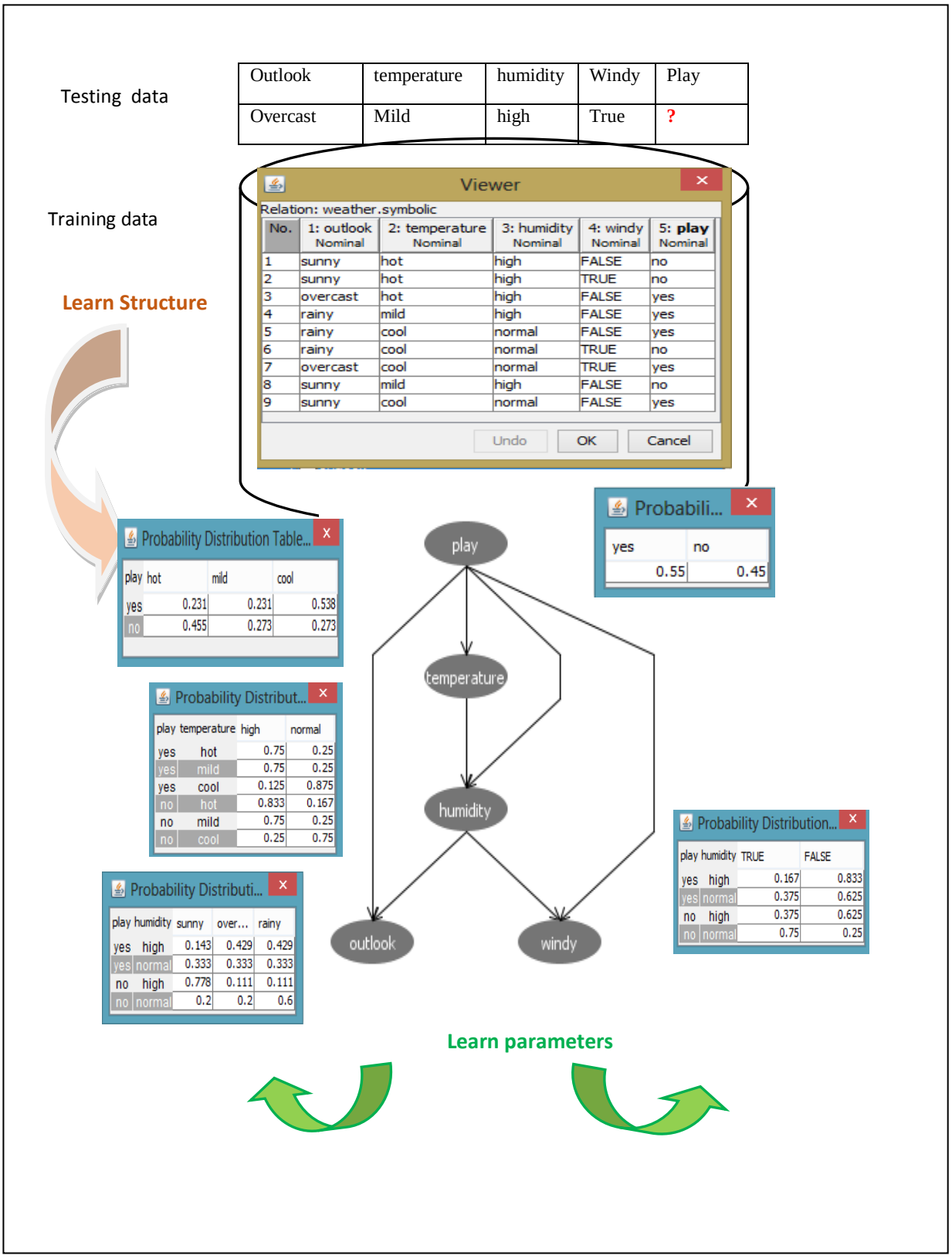


Figure B1.8: Learning structure and parameters of TAN from play-tennis dataset

B2: Example of using a TAN as a classifier for the play-tennis dataset

Figure B2.1 shows how a TAN classifier classifies the testing data to the class label Play={yes, no} that has the highest posterior probability. Clearly, after learning the structure and parameters as shown in Figure B1.8, the TAN classifier can be used to classify each instance in the testing set according to the class label Play={yes, no} that has the highest posterior probability as:

- Testing P(play=yes)
  = P(play=yes) * P(temp=mild| play=yes) *P(humidity= high | temp= mild, play=yes) * P(outlook=overcast| humidity= high, play=yes) * P(wind= true | humidity=high , play=yes).

  = $0.55 * 0.231 * 0.75 * 0.429 * 0.167$ =**0.64**.

- Testing P(play=No)

  = P(play=No) * P(temp=mild| play=No) * P(humidity= high | temp= mild, play=No) * P(outlook=Overcast| humidity= high, play=No) * P(wind= True | humidity=high, play=No)

  = $0.45 * 0.273 * 0.75 * 0.111 * 0.375$ = **0.36.**

  As a result, TAN classifier will classify testing instance to class 'yes' because it has the highest posterior probability as shown in Figure B2.1.

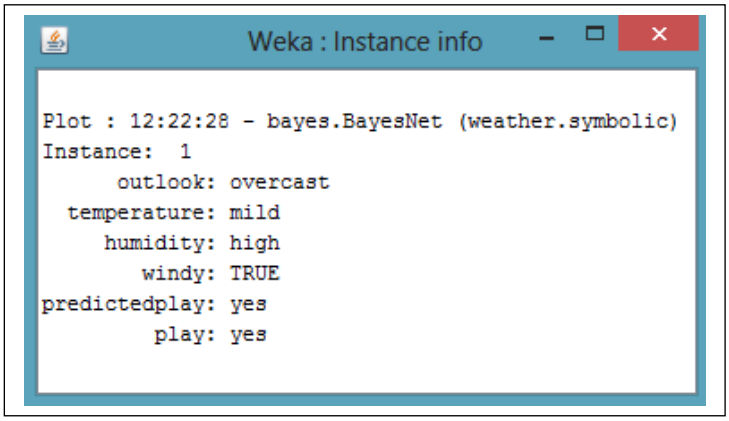


Figure B2.1 : The result of TAN classifier to play-tennis testing data.

Appendix C

C1: Summary of Implementation and Class Diagrams

Class diagram is a graphical way to illustrate the relationships between classes in an object-oriented system. In particular, we used UML (Unified Modelling Language) tool to draw the relationships between classes, where UML is type of diagram that shows the code classes, attributes, methods, and the relationship between the classes. Figure C1.5 shows the class diagram of our code by using *Dia software* to draw the relationships between the classes. In particular, the top level of these classes have been implemented and described as:

Main class : is the class has all the initial values such as costs, and data, and folds. his class is using to call all other classes also, all the results is returned to this class, where this class uses all the following classes:

1. ***Splitting:*** this class is used to splitting a data into two parts; 75% for training, and 25% for testing. As shown in Figure C1.1, the ***Main class*** calls *splitting.splitting_data(Data)* method to split the data into two datasets *Training*, and *Testing* data.

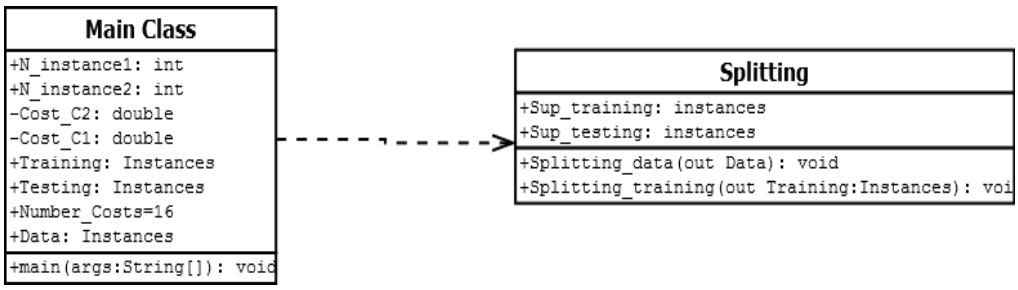


Figure C1.1 : Implementation of class "Splitting"

2. ***Classifier1_CS-BN via Sampling:*** this class is used to implement a Bayesian network by using sampling approach, this class has a class called ***Folk theorem***, which uses to calculate the new training data distribution as described in Chapter 4, in Section 4.1. Where, the ***Main class*** calls these methods:

- *Change_new_distribution(N1,N2)*: this method calls *Sampling(Training, Cost1,Cost2)* method to calculate the new data distribution as given in Chapter 3, equation (3.11). $\frac{N1'}{N2'} = \frac{C1*N1}{C2*N2}$
- *BN_Sampling(Folk_data,Cost1,Cost2)*: this method is used to apply the existing Bayesian network classifier (TAN) on the new distribution (*Folk_data*), to obtain the expected cost and accuracy of that classifier.

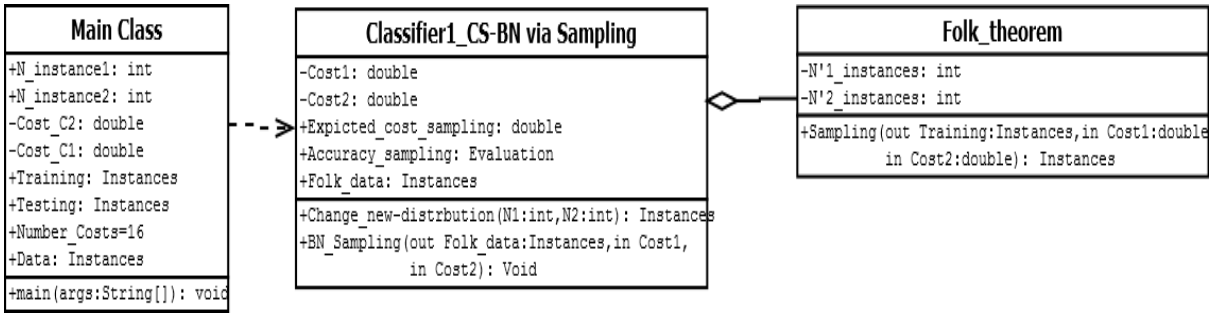


Figure C1.2: Implementation of class "Classifier1_CS-BN via Sampling"

3. ***Classifier2_CS-BN via Amending:*** this class is used to implement our new algorithm that based on amending approach, that described in Section 4.2. Where, the this class uses methods in these classes:

   3.1 Amending_BN_Structuer: it is used to learn the BN structure by calling method *My_BN_Structures()*; where this class is inheritance from original Bayesian network classifier in WEKA (*weka.classifiers.bayes.BayesNet*), but there are some changes on the MDL score function of WEKA class *(weka.classifiers.bayes.search.local.scoreable.MDL)*. Also, there are some changes on class (*weka.classifiers.bayes.search.local.TAN*), these changes are based on calculate cost ratio from class ***Cost_Ratio*** *class*:
   - ***Cost_Ratio :*** it is used to calculate the cost ratio between misclassification costs as described in Chapter 4, equation (4.3) by using method called *Ration(Cost1, Cost2)* $\text{Cost\_ratio}(2) = \frac{\text{cost 1}}{\text{cost 1+ cost 2}}$; $\text{Cost\_ratio}(1) = \frac{\text{cost 2}}{\text{cost 1+ cost 2}}$

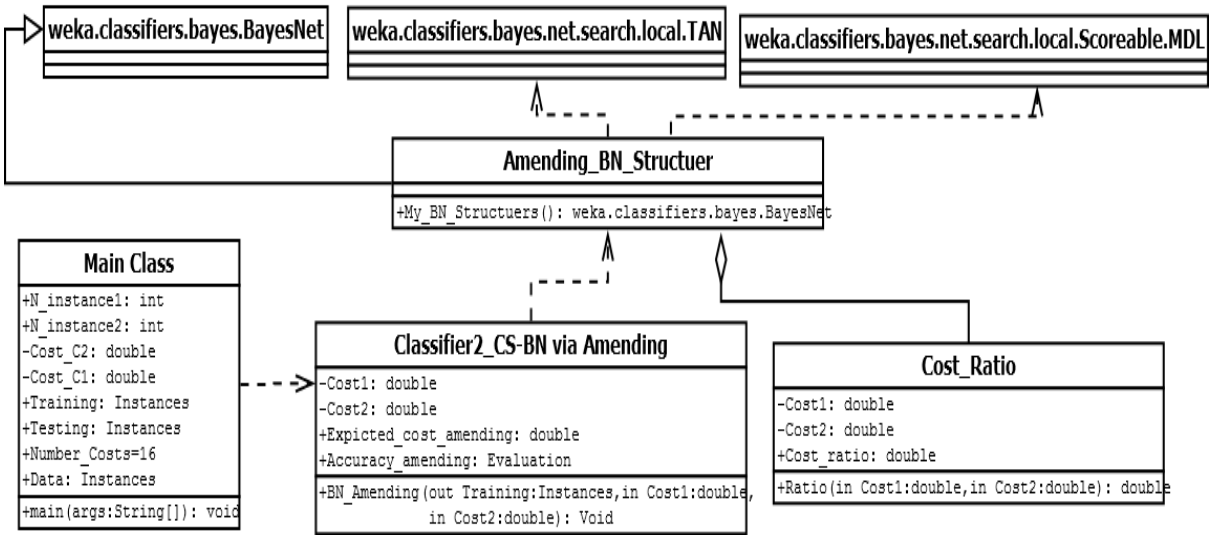


Figure C1.3 : Implementation of class "Amending_BN_Structuer"

3.2 Amending_parameters: it is used to learn the parameters of a BN, where this class is inheritance from original simple estimator class in WEKA (*weka.classifiers.bayes.net.estimate*) but there are some changes on the parameter estimator as presented in equation (4.4). Where the ***Amending_parameters*** class calls method *my_parameter_estimator()* that uses to calculate our new estimator; where this method uses:

- ***Cost_Ratio*** class to calculate the cost ratio for each class label as shown above .

Finally, after learn structure and parameters, the *BN_Amending(Training, Cost1, Cost2)* method used to calculate the last results which are expected cost and accuracy.

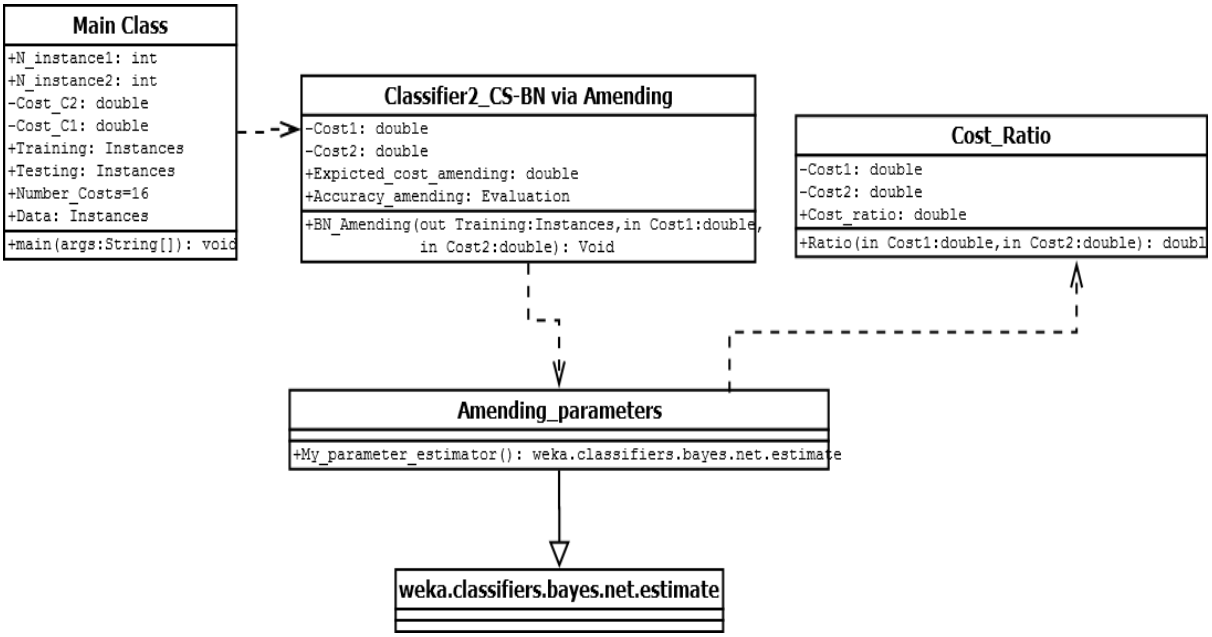


Figure C1.4: Implementation of class "Amending_parameters"

4. ***Classifier3_CS-BN via GA:*** it is used to implement our third classifier by using Genetic algorithm, that described in Section 4.3. First, the ***Main Class*** calls methods called *Splitting_data(Data)*, and *Splitting_training(Tranining)*, in ***Splitting*** class; to split the data into three parts *Sup_training*, *Sup_testing*, and *Testing*. Then, ***Classifier3_CS-BN via GA*** class uses some methods in other classes as:

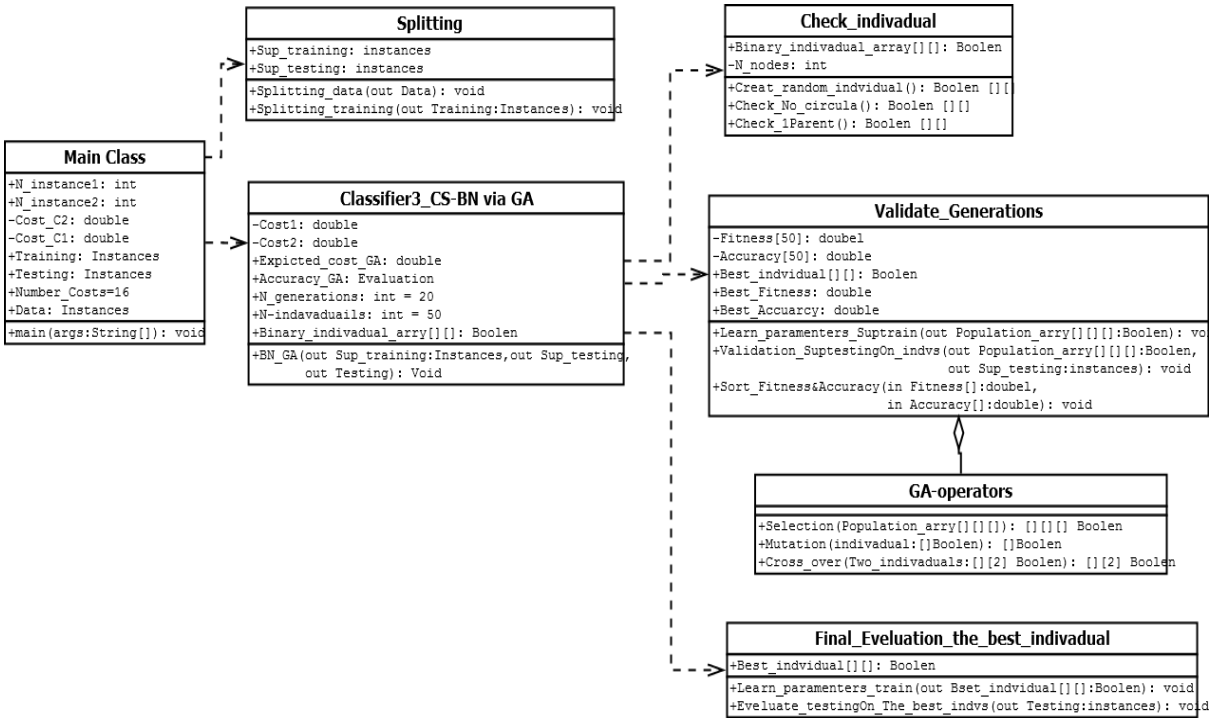


Figure C1.5: Implementation of class "Classifier3_CS-BN via GA"

4.1 Check_indivadual : this class is used to create and check if the generation tree has the same TAN's rules, by using three methods:

- *Creat_random_indvidual():* to generate a random adjacency matrix *Binary_indivadual_array[][]*, as shown in Chapter 4, Figure 4.6(a).
- *Check_No_Circula():* to check if there are no any circulars between nodes that represents as adjacency matrix, as shown in Figure 4.6(b), and Figure 4.6(d).
- *Chack_1Parent():* to check if each node has just 1 parent and other parent (class node) in adjacency matrix.

4.2 Evaluate_Generations: this class is used to learn parameters of BN, and evaluate all individuals in the generation on the fitness function, by using three methods:

- *Learn_parameters_Suptrain(Population_arry[][][]):* this method is used to learn parameters of the learned BN for each individual in a population, by using simple estimator that described in Chapter 2, Section 2.4.2.
- *Validation_SuptestingOn_indvs(Population_arry[][][],Sup_testing):*this method is used to evaluate each TAN (individual) and calculate the expected cost(fitness function), and the accuracy for each tree.
- *Sort_Fitness&Accuracy(Fitness[],Accuracy[]):* this method is used to sort all the individuals according to minimum cost and maximum accuracy .

This class has ***GA-operators*** class, that is used to create the next generation by using three methods:

- *Selection(Population_array[][][]):* it is used to select the best 25 individuals from the previous population in the array, as shown in Figure (4.7).
- *Mutation(indivadual):* used randomly to exchange a bit in the individual.
- *Cross_over(Two_indivaduals[][2]):* it is used randomly to cross over two individuals, as shown in Figure (4.7).

4.3 ***Final_Evaluation_the_best_indivadual:*** this class is used as the last step in this algorithm, to evaluate testing set on the best fitness structure. Where, it uses two method:

- *Learn_parameters_train(Best _individual[][]):* this method is used to learn the parameters of the best Bayesian structure on the *Training* dataset.
- *Evaluate_testingOn_The_best_indvs(Testing):* This method is used to evaluate the best Bayesian structure on the *Testing* dataset to obtain the last results expected cost and accuracy.

5. ***MC+BN:*** it is used to call the existing *MetaCost* classifier based on existing TAN classifier.
6. ***MC+j48:*** it is used to call the existing *MetaCost* classifier based on existing decision tree classifier.
7. ***Original BN:*** it is used to call the original TAN classifier, to find the performance of the algorithm.
8. ***Results_in10_Folds:*** this class is used to collected the performance of all the previous algorithm in 10 folds, and to get the average of each algorithm.
9. ***Write_in excel-sheet:*** it is used as the last stage to write all the results of all algorithms into an excel sheet.

Figure C1.6 shows the classes diagram of top level of all classes that used in our code the classes diagram by using *Dia software.*

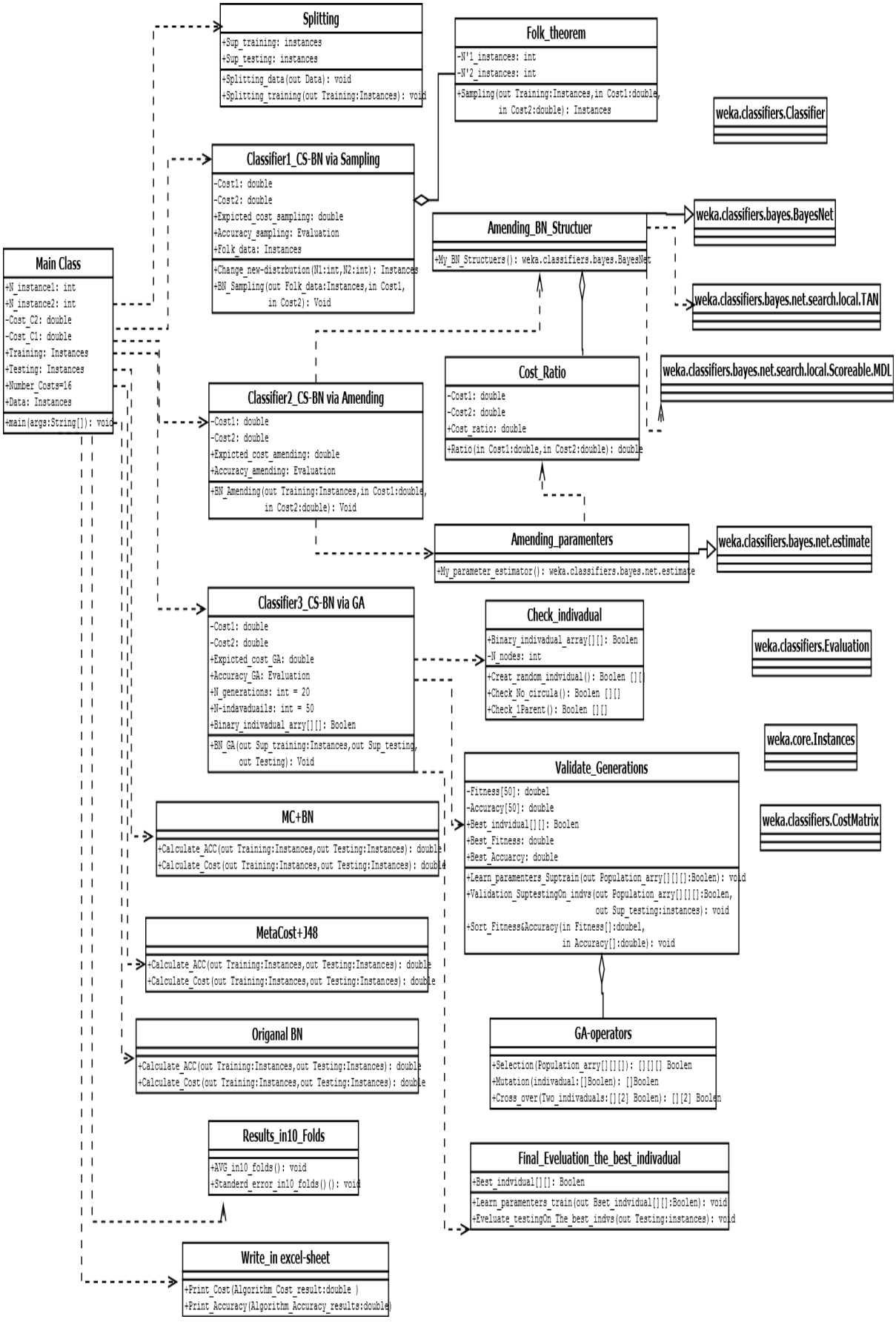


Figure C1.6: The classes diagram by using *Dia software*

C2: Attached DVD, with all the results in excel sheets.