Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering
Mathematical Engineering

Bayesian Networks for Medical Data Analysis

by

Issam Salman

A dissertation thesis submitted to
the Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

Dissertation degree study programme: Mathematical Engineering

Prague 2023

Supervisor:

Ing. Jiří Vomlel, Ph.D.
Institute of Information Theory and Automation
Czech Academy of Sciences
Pod Vodárenskou věží 4
182 00, Prague
Czech Republic

Copyright © 2023 Issam Salman

Abstract and contributions

Nowadays, advances in information technology have revolutionised many disciplines, including medicine and public health. Thanks to these advances, huge amounts of data are generated by individuals every day. Extracting knowledge from large amounts of data poses several challenges, such as how to process these data. In the past decades, data collected from different sources has been neglected due to the lack of efficient tools, and many opportunities to improve patients' knowledge about diseases have been missed. Machine learning is a field of computer science and data analytics research that automates the creation of analytical models. Using algorithms that iteratively learn from data, machine learning allows computers to find hidden insights in data without being explicitly programmed where to look. The iterative aspect of machine learning is important because models are able to adapt themselves when confronted with new data. By using machine learning methods, we can learn from past experience so that we can make reliable, repeatable decisions and achieve results. This science is not new - but it is becoming increasingly important. Because of new computing technologies, machine learning today is nothing like the machine learning of the past. While many machine learning algorithms have been around for a long time, the ability to automatically apply complex mathematical calculations to large amounts of data - over and over again, and faster and faster - is a more recent development.

One of the machine learning methods is Bayesian networks, a type of probabilistic graphical model that uses Bayesian analysis, which is more popular than ever for probability calculations. It has been successfully used in a variety of real-world applications when it comes to supporting decision-making under uncertainty. Bayesian networks aim to model independencies that allow efficient computations, by representing conditional dependencies by edges in a directed graph and then using these conditional dependencies to efficiently compute conditional probabilities in the model (i.e., probabilistic inference). This enables the use of Bayesian networks in applications where it is necessary to model relationships between hundreds of variables. Classifiers based on Bayesian networks are particularly useful. They have a solid theoretical foundation in probability theory and provide competitive predictive performance. There are many algorithms for learning Bayesian network classifiers. In particular, the main contributions of the dissertation thesis are as follows:

1. We implemented a method for building a Tree-Augmented Naive Bayesian (TAN) model and a feature selection method for TAN using incomplete and imbalanced data - Selective TAN (STAN).
2. We analysed medical records of patients suffering from acute myocardial infarction (AMI) from the third world country Syria and a developed country - the Czech Republic.
3. We applied machine learning methods to predict myocardial infarction mortality.
4. We designed a methodology for learning the structure of Bayesian networks and Belief Noisy-Or models from incomplete datasets.

Keywords:
Machine Learning, Data mining, Data analysis, Classification, Bayesian networks, Belief Noisy-Or, Structure learning, Acute Myocardial Infarction.

Acknowledgements

First of all, I would like to express my gratitude to my dissertation thesis supervisor, Ing. Jiří Vomlel, Ph.D.. He has been a constant source of encouragement and insight during my research and helped me with numerous problems and professional advancements.

Special thanks go to the staff of the Faculty of Nuclear Sciences and Physical Engineering for providing a pleasant and flexible environment for my research. I would especially like to thank the management of the faculty for providing most of the funding for my research. My research was also partly supported by by the Grant Agency of the Czech Technical University in Prague, Grant No. SGS16/253/OHK3/3T/14 and SGS20/132/OHK4/2T/14.

Above all, I would like to thank my wife Zeina for her love and constant support, for all the late nights and early mornings, and for keeping me sane these past few months. Thank you for being my muse and sounding board. But most of all, thank you for being my best friend.

Finally, my greatest thanks go to my parents, brothers and sons Jad and Ward for their endless patience, support and unwavering belief in me and their care. I owe you all everything and without you, I would not be the person I am today.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

δ	Standard Delta Function
$\mathbb{B}_{X_j}$	Markov blanket of X_j
$\mathbf{U}$	Set of Random Variables
$\mathbf{u}$	Vector of Values
$\mathcal{D}$	Dataset
$\mathcal{L}$	Logistic Regression Function
$\mathcal{N}(\mathbf{u})$	Subset of Indices of Available Variables in $\mathbf{u}$
$\mathcal{X}$	Finite Set of States of Variable X
Π_{X_i}	Parents set of X_i
θ	Set of Parameters
$F()$	Joint Probability Distribution Function
$f()$	Conditional Probability Distribution Function
G	A Directed Acyclic Graph
$L()$	Likelihood Function
$LL()$	Log-likelihood Function
MDL, BIC	Scoring Functions of G
$N(X_i)$	Neighbors of $X_i \in G$
$P()$	Product Distribution Mixtures

$s(G)$	Score Value of G
w	Probabilistic Weight
$W()$	Conditional Component Weights
$X, Z, Y..$	Random Variables

Chapter 1

Introduction

This chapter begins with a brief overview of machine learning approaches and their role in healthcare. We explain several motivations for addressing the problems of scarce data and learning the model with a small amount of labelled data. We discuss the most common methods used in this work and provide intuitive explanations and key insights into why the proposed methods work under scarcity conditions. Furthermore, we then present the motivation, importance of the topic, problem statement, methodology, and research contributions of these studies. Finally, the structure of the thesis is discussed.

1.1 Motivation

An enormous amount of data is generated every day, and analysing this data is impractical without the help of automated procedures. Machine learning [1] provides these procedures. The most commonly used form of machine learning is supervised classification [2]. Its goal is to learn a mapping from the descriptive features of an object to the set of possible classes, given a set of feature-class pairs. Probabilities play a central role in modern machine learning [3]. Probabilistic Graphical Models (PGMs) [4] have emerged as a general framework for describing and applying probabilistic models. A PGM allows us to efficiently encode a joint distribution over a set of random variables by making conditional independence assumptions.

Bayesian Networks (BNs) have been used in many applications. The difficulty of learning a BN can be divided into two categories: (1) structural learning, which involves determining the topology of the network, and (2) parametric learning, which involves computing the conditional probability tables (CPTs) for a given network. By far the most challenging of the two is learning the structure of a BN. The structure is made up of nodes representing the variables, and edges between pairs of nodes indicating the conditional probability for the associated random variables. CPTs measure the relationships between variables. However, it has been pointed out that it is usually difficult to quantify the CPTs due to their complexity. One of the most appropriate solutions to this problem is the Belief Noisy-OR (NOR) [5].

For example, in medical diagnosis, one may ask about the probability of a particular disease if a patient has certain symptoms. These diagnostic problems are often complex and involve many interrelated variables. There may be many symptoms and even more possible causes. In practice, it is usually only possible to obtain the inverse conditional probability, i.e. the probability of evidence given the cause, the probability of observing symptoms if the patient has the disease. In these cases, a Bayesian approach is appropriate, and Bayesian networks, or alternatively graphical models, are very useful tools not only for dealing with uncertainty, but also for dealing with complexity and, more importantly, for modelling causality [6]. Bayesian networks have already found application in health outcomes research and medical decision analysis, but modelling random events and their probability distributions can be equally helpful in health economics or public health research.

1.2 Problem Statement

The prevalence of chronic diseases is increasing worldwide, and their management is one of the greatest challenges facing healthcare systems. As a result, healthcare systems are seeking better solutions to improve quality, efficiency, and reduce the cost of care. In general, healthcare institutions are becoming increasingly dependent on advances in technology, and the use of machine learning (ML) techniques can provide valuable support to assist physicians in a variety of ways.

Over the past decade, ML has attracted attention from various fields, including healthcare, with the aim of improving service quality and care. To date, advanced ML techniques in healthcare have focused on solving prognostic problems, including those in mental health [7] and human behaviour [8]. It is often argued that the use of ML tools in medicine will lead to improvements in patient care, offering opportunities to enhance the work of physicians, including the efficiency and quality of healthcare [9].

Current technologies generate and collect large amounts of data, making it too complex to analyse using traditional methods. Building such systems presents a number of challenges, with particular interest in building robust learning systems that work in real-world environments. However, collecting patient data poses several difficulties, including incompleteness (missing label values), noise in the dataset, irrelevant feature selection, and data scarcity due to the small number of available patient records. This research addresses the problems of learning Bayesian network methods from incomplete, imbalanced, and noisy data.

1.3 Related Work

Several machine learning methods have been applied to medical problems. Bayesian networks have also found applications in this field. However, there is still room for improvement, especially in learning Bayesian networks and Belief Noisy-Or (BNO) models from incomplete and imbalanced data. In the next chapter, we will review previous results and related work.

1.4 Goals of the Dissertation Thesis

1. Learning the structure of Bayesian networks and Belief Noisy-Or models from incomplete datasets.
2. Dealing with incomplete and imbalanced data for Chow-Liu, tree-augmented naive Bayesian (TAN), and selective TAN (STAN), which is a feature selection method for TAN.
3. An application of machine learning methods for heart attack mortality prediction based on their features' values.

1.5 Structure of the Dissertation Thesis

The thesis is organized into following chapters:

1. **Introduction**: Bayesian Network (BN) models can help us understand our environment and discover the "laws" of nature in the sciences: biology, genetics, chemistry, and even physics. Today, practical systems along the same lines can and do help doctors narrow down their diagnostic choices for diseases. In this capacity, automatic or even semi-automatic construction of models can be invaluable.
2. **Background and State-of-the-Art**: Several machine learning methods have been applied to medical problems, including Bayesian Networks (BNs). However, to the best of our knowledge, further improvements are needed, such as learning BNs and Bayesian network structures (BNOs) from incomplete data.
3. **Overview of Our Approach**: We provide an approach to learn the optimal BN structure from incomplete data by adapting [10]. This adaptation imputes missing values using mixtures of Gaussian distributions learned by the EM algorithm [11]. We have shown that the sequence of log-likelihood values generated by the E-step and M-step of the EM algorithm is non-decreasing, and that the algorithm converges. We reduce the collection of candidate parent sets for a variable, which can speed up the learning algorithm.
4. **Main Results**: We have empirically shown that our approach performs better than other tested algorithms on several studied BNs and in different scenarios.
5. **Conclusions**: Based on these experiments, we can recommend this algorithm for practitioners who use BNs or BNOs with incomplete data.

6. **Future Work**: An interesting topic for future research might be learning the structure of large BNO networks from incomplete data.

Chapter 2

Background

2.1 Why Bayesian Network Models

In this research, we focus on a special class of probabilistic graphical models called Bayesian Networks (BNs). BNs can be viewed as a generally accepted formalism for representing and efficiently reasoning with uncertain information [12]. A BN provides a model of conditional independence using a directed acyclic graph.

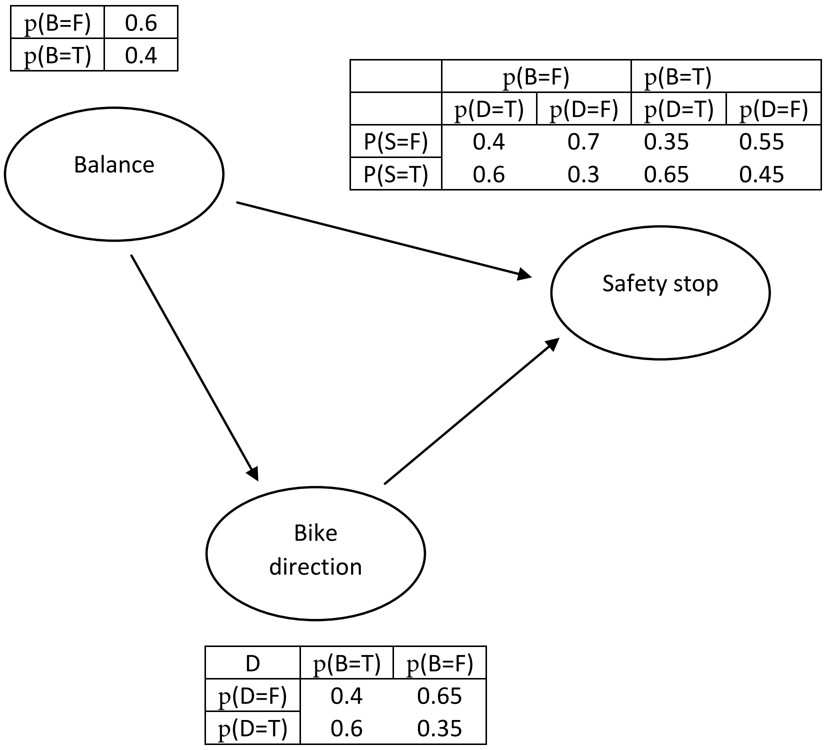
There are numerous reasons for our choice. First, we need a specific class of models to illustrate and apply our concepts and to test the resulting algorithms. Second, we intend to use probability theory as a foundation. Probability theory is an ancient, well-established theory that has withstood the test of time and has become one of the cornerstones of science. Our use of probability theory is born out of necessity, as most areas of AI technology involve uncertainty that we must address with clarity and rationality from the outset.

While various models can be used to describe uncertain domains, such as decision trees, artificial neural networks, mixtures of Gaussian distributions, Markov networks, etc., only in the Bayesian network literature do we find claims of the ability to represent and learn directed causal relationships. In short, the reasons for choosing Bayesian networks are as follows:

- They are graphical models, capable of representing relationships clearly and intuitively.
- They are directional, thus capable of representing cause-effect relationships.
- They can handle uncertainty.
- They can be used to represent both indirect and direct causation.

Figure 2.1 briefly illustrates the dual nature of BN models, highlighting their ability to represent both causal relationships and joint probability distributions. In this representation, nodes symbolize events (variables), while edges depict direct effects (e.g., the variable

"Balance" has a direct effect on the variable "Bike direction," meaning that losing balance can result in a loss of direction). Each table within the figure provides the probability of a single variable (e.g., the table associated with the node "Balance") or the conditional probability of an event given the state of another variable (e.g., the probability of "Balance" and "Bike direction"). For instance, the probability of maintaining direction given that balance is maintained is 0.6, or 60



p(B=F)	0.6
p(B=T)	0.4

	p(B=F)		p(B=T)	
	p(D=T)	p(D=F)	p(D=T)	p(D=F)
P(S=F)	0.4	0.7	0.35	0.55
P(S=T)	0.6	0.3	0.65	0.45

D	p(B=T)	p(B=F)
p(D=F)	0.4	0.65
p(D=T)	0.6	0.35

Figure 2.1: An example BN that can be used for modeling riding a bike.

2.2 Bayesian Networks and Health Care

The formalism of Bayesian networks is used to specify a joint probability distribution over a collection of random variables. Consequently, a Bayesian network primarily serves probabilistic reasoning, such as diagnosing a particular patient and predicting the effects of a treatment. When it comes to decision-making, like choosing the best treatment alternative for a specific patient, the network formalism may not be directly applicable. However, contemplating treatment alternatives involves considering the expected effects of various options, thereby encompassing both diagnostic and, more crucially, prognostic reasoning. To facilitate the selection of an optimal treatment, Bayesian networks and their associated algorithms are often integrated into decision support systems, offering the necessary constructs from decision theory for selecting an optimal treatment based on predictions [13]. Alternatively, the Bayesian network formalism can be extended to incorporate knowledge about decisions and preferences. An example of such an extended formalism is the influence diagram formalism [14]. Similar to a Bayesian network, an influence diagram consists of an acyclic directed graph. In this graph, nodes are divided into three categories: probabilistic nodes representing random variables, decision nodes modelling different treatment alternatives, and a value node modelling respective preferences. Influence diagrams for treatment selection also exhibit a clear and generalized structure.

Given that the topology of a Bayesian network can be interpreted as a representation of uncertain interactions between variables, there is a growing interest in bioinformatics in utilizing Bayesian networks to decipher molecular mechanisms at the cellular level. For example, tracing interactions between genes based on experimentally obtained expression data in microarrays is currently a significant research topic [15]. Biological data are often collected over time, and analysing temporal patterns can reveal how variables interact as a function of time. This is a common task in molecular biology, and Bayesian networks are increasingly being used to analyse such biological time series data [16].

2.3 Probability Distribution Represented by a Bayesian Network

A Bayesian network encodes a joint probability distribution over a set of random variables $\mathbf{U} = \{X_1, X_2, \ldots, X_m\}$. We consider only discrete variables in this work, which is the most common current usage of Bayesian networks. A finite set of states of a variable X_i will be denoted by $\mathcal{X}_i$.

Conditional probability distributions (CPDs) are attached to each variable in the network. Their purpose is to quantify the strength of the relationships depicted in the Bayesian network through its structure: these CPDs mathematically describe the behavior of that variable under every possible value assignment of its parents. Since specifying this behavior requires a number of parameters exponential in the number of parents, and since this number is typically smaller than the number of variables in the domain, this approach results in exponential savings in space and time.

Formally, a Bayesian network for $\mathbf{U}$ is a pair $B = \langle G, \theta \rangle$. Its first component, G, is a directed acyclic graph whose vertices correspond to the set of random variables $\mathbf{U}$, and whose edges represent direct dependencies between these variables. The graph G encodes independence assumptions: each variable X_i is independent of its non-descendants given its parents in G.

The second component of the pair, θ, represents the set of parameters that quantify the network. It contains parameter $\theta_{x_i|\Pi_{x_i}} = f(x_i|\Pi_{x_i})$ for each possible value x_i of X_i and Π_{x_i} of Π_{X_i}, where Π_{X_i} denotes the set of parents of X_i in G and $f(x_i|\Pi_{x_i})$ is the conditional probability distribution.

Accordingly, a Bayesian network B defines a unique joint probability distribution F over $\mathbf{U}$ given by:

$$F(X_1 = x_1, \ldots, X_m = x_m) = \prod_{i=1}^{m} f(X_i = x_i | \Pi_{X_i} = \Pi_{x_i}) = \prod_{i=1}^{m} \theta_{x_i|\Pi_{x_i}} \tag{2.1}$$

for each $\Pi_{X_i=x_i}$ which is a parent of $X_i = x_i$, where f is conditional probability distribution.

2.3.1 D-separation

Unlike regular graph connectivity concepts, conditioning on a node can "block" or "unblock" a dependency path between two nodes, depending on the direction of traversal of that node along that path [17, 18].

Let's assume that a subset of nodes $\mathbf{Z}$ lies on an undirected path p between X and Y, and that p is a path between X and Y in the graph G (for simplicity). We want to determine whether a path p from X to Y is blocked by $\mathbf{Z}$. According to the d-separation, the path is blocked by $\mathbf{Z}$ if:

- X, Y, and Z_i are connected nodes where $Z_i \in \mathbf{Z}$ is in the chain between X and Y ($X \leftrightarrow \cdots \leftrightarrow Z_i \cdots \leftrightarrow Y$), or
- X and Y are connected by a common cause $Z_i \in \mathbf{Z}$ (i.e., X and Y are children/descendants of Z_i, here: $X \cdots \leftarrow Z_i \rightarrow \ldots Y$), or
- X and Y are connected by a common effect ('collider'), but $Z_i \in \mathbf{Z}$ is not that common effect, and Z_i is not one of the effects of the common effect.

When influence can flow from X to Y via Z_i, we say that the trail $X \leftrightarrow Z_i \leftrightarrow Y$ is active. Our results for active two-edge trails were summarized into four classes:

- Causal trail (showed the strongest effects) $X \rightarrow Z_i \rightarrow Y$: active if and only if Z_i is not observed.
- Evidential trail (showed a correlation) $X \leftarrow Z_i \leftarrow Y$: active if and only if Z_i is not observed.
- Common cause $X \leftarrow Z_i \rightarrow Y$: active if and only if Z_i is not observed.
- Common effect $X \rightarrow Z_i \leftarrow Y$: active if and only if either Z_i or one of Z_i's descendants is observed.

Consider the case of a longer path $X_1 \leftrightarrow \cdots \leftrightarrow X_n$. For influence to flow from X_1 to X_n, it needs to flow through every single node on the path. In other words, the path $X_1 \leftrightarrow \cdots \leftrightarrow X_n$ is active given $\mathbf{Z}$ if there is a common effect $X \rightarrow Z_i \leftarrow Y$, where Z_i or one of its descendants is in $\mathbf{Z}$, and no other node along the trail is in $\mathbf{Z}$. Also, X_1 and X_n are not in $\mathbf{Z}$. For example, $E \rightarrow F \leftarrow I \rightarrow S$ is not an active trail for $\mathbf{Z} = \varnothing$ because the common effect $E \rightarrow F \leftarrow I$ is not activated. That same trail is active when $\mathbf{Z} = \{L\}$ because observing the descendant of F activates the common effect. On the other hand, when $\mathbf{Z} = \{L, I\}$, the trail is not active because observing I blocks the trail $F \leftarrow I \rightarrow S$.

More generally, what about graphs where there is more than one path between two nodes? For example, is X independent of Y given the set of nodes $\mathbf{Z}$, i.e.,

$$\text{d-sep}_G(X \perp Y|\mathbf{Z}) = \text{Yes}$$

in Figure 2.2?

Our flow intuition continues to carry through: one node can influence another if there is any trail along which influence can flow. Putting these intuitions together, we obtain the notion of d-separation, which provides us with a notion of separation between nodes in a directed graph.

****Definition** 2.1.** D-separation: Let $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ be subsets of nodes in DAG (Directed Acyclic Graph) G. We write

$$\text{d-sep}_G(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z}) = \text{Yes}$$

if and only if there is no active trail between any node $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$ given the set of nodes $\mathbf{Z}$ in the graph G. This indicates that the variables in $\mathbf{X}$ are independent of the variables in $\mathbf{Y}$ given $\mathbf{Z}$ in the context of the directed acyclic graph G.

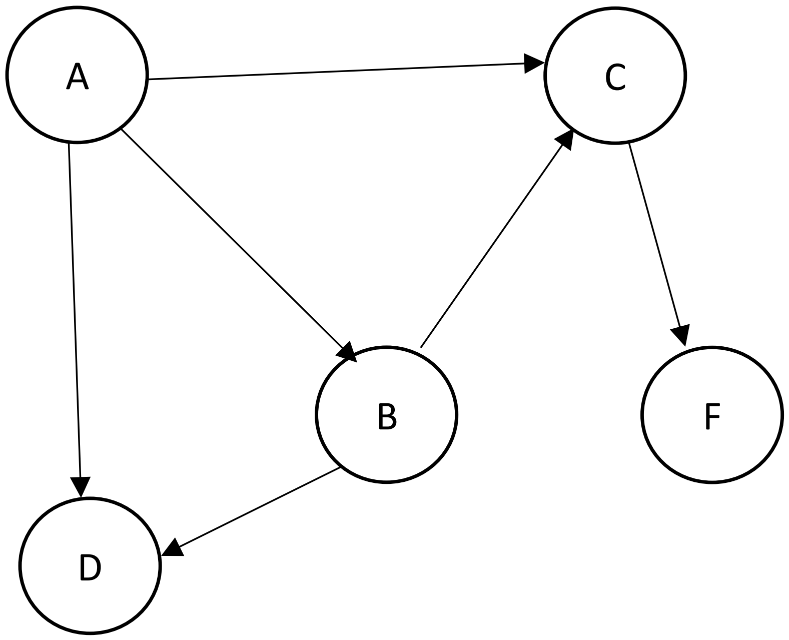


Figure 2.2: Example of d-separation d-sep$_G(D \perp F|C)$=Yes, d-sep$_G(C \perp D|B)$= No, and d-sep$_G(C \perp D|\{A, B\})$=Yes.

2.4 Assumptions for Learning the Causal Structure

As mentioned earlier, the Bayesian Network (BN) model encodes a set of dependencies that exist in a domain.

Causal Sufficiency Assumptions: There are no common unobserved (even as hidden or latent) variables in the domain that are parents of one or more observed variables in the domain.

Markov Assumptions: Any node in a Bayesian network, given its parents, is conditionally independent of its non-descendants. In simpler terms, it is assumed that a node depends only on its directed parents.

The Causal Sufficiency Assumptions state that there are no unobserved nodes in the domain that might describe the dependencies observed in the data or their absence. This assumption is crucial for applications that require discovering the true underlying causal structure of the domain. However, it is often challenging to uphold, as it is relatively easy to envision another node, perhaps at a different level of detail, that can be added to the model as an endogenous or even exogenous node. By applying the Markov Assumptions and a set of assumptions described in [5], one can derive the complete set of independence relations implied by the BN model. It's important to note that the presence of an edge or an unblocked path between two nodes does not necessarily imply that these nodes are dependent.

Faithfulness Assumptions: Let G be a Bayesian network graph. The joint probability distribution F is exactly that which results from d-separation in the corresponding causal graph G. Combined with the causal Markov assumption, this implies that G is a perfect representation of F.

2.5 Bayesian Network Structure Learning

Note that a Bayesian Network (BN) can be viewed from two perspectives: as an effective coding of an independence relationship and as an effective encoding of a high-dimensional distribution of probabilities. One option for learning the structure is to rely on specialists in the field through a deliberate and meticulous process of knowledge gathering. This involves training experts in probabilistic graphical modelling, validating expert opinions, and extracting and testing information. This process all too often leads to disagreements among experts and a lack of reliability in the model. Nonetheless, in many fields where data is scarce, this is one of the key approaches to model building.

Another mechanism is the automatic derivation of the model based on a dataset. We adopt a machine learning approach (ML) for this purpose, in order to avoid the vast field of human knowledge acquisition. For a dataset $\mathcal{D} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$, where $\mathbf{u}_i$ is a vector of values representing an instantiation of all variables in $\mathbf{U}$, Bayesian network (BN) structure learning is the problem of inferring a network structure from $\mathcal{D}$. Let's assume that $\mathcal{D}$ is complete and discrete. Consequently, the task of finding the optimal Bayesian network reduces to finding the optimal structure.

The optimal structure can be learned to use three approaches in ML. The first is the constraint-based approach to structure learning, which attempts to reconstruct a Bayesian network by analysing data independence. The second is the score-based approach, which looks for Bayesian networks that adequately describe the available data with the best score. The core of the approach is to assign a score value $s(G)$ to each acyclic directed graph G. The score function defines an overall order (up to equivalences) over the structures in such a way that structures with a better description of the data are assigned a higher value. The last approach is a hybrid approach, which combines the two previous approaches.

2.5.1 Trees

A directed acyclic graph on $\{X_1, X_2, \ldots, X_m\}$ is a tree if Π_{X_i} contains exactly one parent for all X_i, except for one variable that has no parents (this variable is referred to as the root). A tree network can be described by identifying the parent of each variable [5].

A function $\pi : \{1, \ldots, m\} \to \{0, \ldots, m\}$ is said to define a tree over $X_1, X_2, \ldots, X_m$ if there is exactly one i such that $\pi(i) = 0$ (namely the root of the tree), and there is no sequence $i_1, \ldots, i_k$ such that $\pi(i_j) = i_{j+1}$ for $i \leq j < k$ and $\pi(i_k) = i_1$ (i.e., no cycles). Such a function defines a tree network where $\Pi_{X_i} = \{X_{\pi(i)}\}$ if $\pi(i) > 0$ and $\Pi_{X_i} = \emptyset$ if $\pi(i) = 0$.

2.5.2 Maximum Likelihood TAN

Let $\{X_1, X_2, \ldots, X_m\}$ be a set of attribute variables, and let C be the class variable. We say that Bayesian network is a TAN model (B_T) if $\Pi_C = \emptyset$, where Π_C is the parent set of C, and there is a function π that defines a tree over $\{X_1, X_2, \ldots, X_m\}$ with an edge from C to each node. The optimization problem consists of finding a tree-defining function π over $\{X_1, X_2, \ldots, X_m\}$ such that the log-likelihood is maximized [12]:

$$LL(B_T|\mathcal{D}, \theta) = \sum_{\mathbf{u} \in \mathcal{D}} \log L(\mathbf{u}, \theta)$$

Here, $\mathbf{u}$ represents a vector of values, and the summation is taken over all possible vectors $\mathbf{u}$ that are part of the dataset $\mathcal{D}$. The function $L(\mathbf{u}, \theta)$ is the likelihood function for the TAN model, and θ represents the model parameters.

To learn the maximum likelihood TAN, we should use the following equation to compute the parameters [12]:

$$\theta_{X_i=x_i|\Pi_{X_i=x_i}} = \frac{N_{X_i=x_i,\Pi_{X_i=x_i}}(X_i = x_i, \Pi_{X_i=x_i})}{N_{\Pi_{X_i=x_i}}(\Pi_{X_i=x_i})}$$

Here, $\theta_{X_i=x_i|\Pi_{X_i=x_i}}$ represents the parameter for attribute $X_i = x_i$ given its parent set $\Pi_{X_i=x_i}$. $N_{X_i=x_i,\Pi_{X_i=x_i}}(X_i = x_i, \Pi_{X_i=x_i})$ represents the number of times that attribute X_i has value x_i, and its parents have values Π_{x_i} in the dataset. Similarly, $N_{\Pi_{X_i=x_i}}(\Pi_{X_i=x_i})$ is the number of times that the parents of attribute X_i have values $\Pi_{X_i=x_i}$ in the dataset.

2.5.3 Constraint-Based Vs. Score-Based

- **Constraint-Based Structure Learning:** This approach aims to recover the independence relationships in the data by constructing a Bayesian network structure that is consistent with the observed dependencies/independencies implicit in the data. A notable advantage of this method over search and evaluation methods is its efficiency in reconstructing sparsely populated networks - those that are not densely connected. It also provides more accurate independence test results, even when the sample size is arbitrary [19]. However, because these algorithms rely on statistical tests to determine whether an arc between two variables should be included, they are sensitive to the amount of data. The reliability of these tests depends on both the size of the database and the number of variables in the conditioning set. In other words, these algorithms require a large amount of data to learn independence with certainty, and high-order independence tests may be unreliable unless the sample size is substantial [20].

- **Score-based structure learning:** This approach defines a metric (score) to assess how well the dependencies/independencies represented by a given Bayesian network structure fit the data. A search algorithm then searches for structures that maximise or minimise this score. A major advantage of these algorithms is that they do not require the setting of a statistical threshold. They also inherently follow the philosophy of Occam's razor, favouring simpler models over more complex ones in their evaluation criteria. However, a notable drawback is their computational intractability [21]. To mitigate this problem (i.e. the large search space), some algorithms, such as K2 [22], assume an order of ancestral nodes, which means that variables are arranged in a list where a variable on the left may be a parent of a variable on the right, but not vice versa. Another important drawback of most score-based algorithms is their heuristic nature, which makes it impossible to find the best network structure, only a good local one. As a result, these algorithms may need to be run several times to avoid getting stuck in a local maximum, which can be very time consuming. However, GOBNILP [23] has addressed some of these problems by guaranteeing an optimal solution.

2.5.4 Score-Based

Score-based learning is a commonly used technique for determining the optimal structure of Bayesian networks. In this approach, each candidate structure is assigned a Bayesian Network (BN) score to measure its goodness of fit to the data. The goal of score-based learning is to find the structure that maximizes this score, which typically indicates how well the BN describes the data set $\mathcal{D}$.

In essence, the goal is to find an acyclic graph G that maximizes the score function $L(G, \theta) \to \mathbb{R}$ (real numbers) where L is likelihood function. Given a structure G, its score is defined as

$$s(G|\mathcal{D}) = \log(L(\mathcal{D}|G, \theta)) \tag{2.2}$$

Here, θ represents a vector of model parameters.

The learning problem is to find G^*, where:

$$G^* = \arg\max_G s(G|\mathcal{D}) \tag{2.3}$$

In other words, we want to find a graph that maximizes the posterior probability of the data set $\mathcal{D}$ given G. Score-based algorithms are designed to maximize this score. This calculation can be transformed into a more convenient form using Bayes' rule [24]:

$$L(G|\mathcal{D},\theta) = \frac{L(\mathcal{D}|G,\theta)L(G,\theta)}{L(\mathcal{D},\theta)} \tag{2.4}$$

To maximize this value, we only need to maximize the numerator, as the denominator does not depend on G:

$$s(G|\mathcal{D}) = \log(L(\mathcal{D}|G,\theta)) + \log(L(G,\theta)) \tag{2.5}$$

Many scoring functions are expressed as penalized log-likelihood (LL) functions. LL represents the logarithmic probability of $\mathcal{D}$ given G and can be calculated as

$$LL(G|\mathcal{D},\theta) = \log(L(G|\mathcal{D},\theta)) \tag{2.6}$$

Adding an arc to a mesh does not reduce its probability. In fact, the extra arc should be ignored if it doesn't add any information. In addition, additional arcs can lead to overfitting on training data and increase the runtime of downstream analyses such as inference and prediction.

To address these issues, penalized LL functions aim to penalize complex networks. These functions typically take the form of decomposable penalized LL (DPLL) values:

$$DPLL(G,\mathcal{D}) = LL(G|\mathcal{D},\theta) - \sum_{i=1}^{m} \text{Penalty}(X_i, G, D) \tag{2.7}$$

There are several well-known DPLL scoring functions for learning Bayesian networks. One commonly used scoring function is based on the principle of minimal description length (MDL) [25]. The MDL approach treats the scoring of Bayesian networks as an information-theoretic task, where the data is minimally encoded into the network structure and the unexplained data.

****Definition** 2.2.** Let $B = \langle G, \theta \rangle$ be a Bayesian network, and let $\mathcal{D} = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ be a training set, where each $\mathbf{u}_i$ is a vector of values of all variables in $\mathbf{U} = \{X_1, X_2, \ldots, X_m\}$. The MDL scoring function of a network B given a training data set $\mathcal{D}$, written as $MDL(B|\mathcal{D})$, is defined as

$$MDL(G|\mathcal{D}) = LL(G|\mathcal{D},\theta) - \frac{\log n}{2}|G| \tag{2.8}$$

Here $|G|$ represents the number of parameters in the network. The first term measures model fit, and the second term penalizes model complexity.

The penalty term for MDL is usually larger than for most other scoring functions. Networks optimized using MDL tend to minimize the scoring function rather than maximize it.

The Bayesian Information Criterion (BIC) [26] is another scoring function that is equivalent to MDL for Bayesian networks but is derived based on the asymptotic behavior of the models. If the score is decomposable, it can be written as the sum of the scores for each variable and its parent set:

$$BIC(G|\mathcal{D}) = \sum_{i=1}^{m} BIC(X_i|\Pi_{X_i}) \tag{2.9}$$

Here $BIC(X_i|\Pi_{X_i}) = LL(X_i|\Pi_{X_i}, \theta) - \text{Penalty}(X_i|\Pi_{X_i})$.

Score-based algorithms aim to optimize this score and return the structure G that maximizes it. However, since the space of all possible structures is at least exponential in the number of variables m, this poses several challenges. A popular choice for exploring this space is hill climbing [27], an iterative algorithm that incrementally improves an initial solution until no further improvements can be found. An example of hill climbing is illustrated in Figure 2.3.

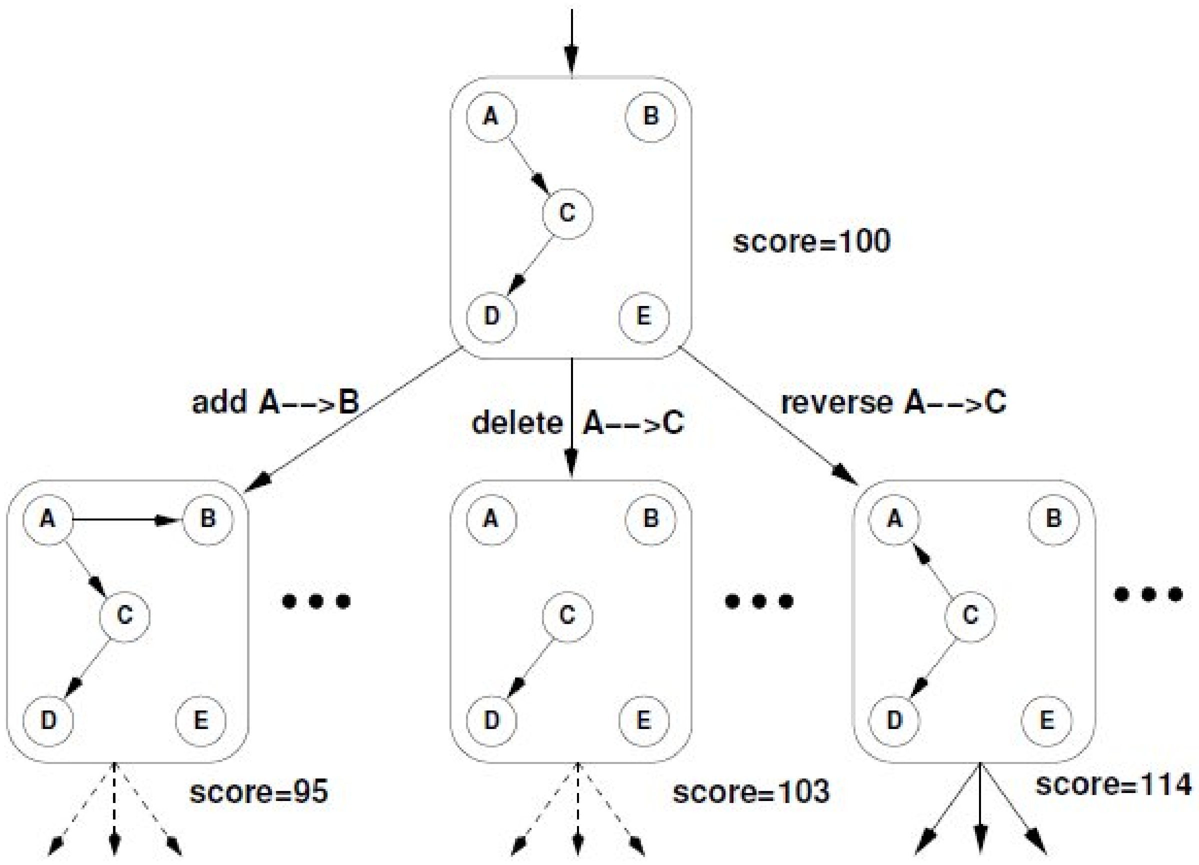


Figure 2.3: Illustration of a BN structure hill-climbing search procedure.

2.5.5 Constraint-Based

Another way to learn the structure of a Bayesian network (BN) is through the use of constraints. These constraints typically involve conditional independence statements (e.g. see [28]). The conditional independence tests used in practice are statistical tests on the data set. In order to use the results to reconstruct structure, several assumptions must be made: Causal Sufficiency, Causal Markov, and Faithfulness (see [29, 30]).

With these assumptions, the presence of an edge between two variables or the direction of that edge can be determined, although the latter is only possible in certain cases. All constraint-based structure learning algorithms share a common three-phase structure, inherited from the IC algorithm [31] through the PC algorithm [32] and the GS algorithm [33].

****Definition** 2.3.**: Let X_k be a random variable in the set $\mathbf{U} = \{X_1, \ldots, X_m\}$. The Markov Blanket of a node X_k is the set containing the parents, children and co-parents of the node, where the co-parents of a node are the parents of its children. Figure 2.4 shows an example of a Markov Blanket in a Bayesian network.

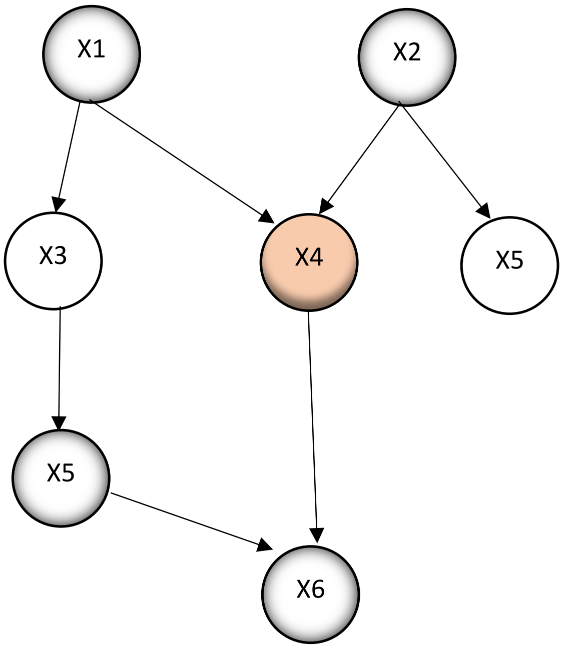


Figure 2.4: Example of a Markov blanket of variable X_4. The members of the blanket are shown shaded.

The first, optional, phase consists of learning the Markov blanket of each node to reduce the number of candidate DAGs early on. Any algorithm for learning Markov blankets can be plugged in as the first step (learning Markov blankets for each variable) and extended into a full BN structure learning algorithm, as originally suggested in [17] for the GS algorithm. Once all Markov blankets have been learned, they are checked for consistency in step 2 - check whether the Markov blankets are symmetric); by definition, $X_i \in \Pi_{X_j} \Leftrightarrow X_j \in \mathbb{B}_{X_i}$, where $\mathbb{B}_{X_j}$ is the Markov blanket of X_j. Asymmetries are corrected by treating them as false positives and removing the offending nodes from each other's Markov blankets.

The second phase learns the skeleton of the DAG, i.e., it identifies which arcs are present in the DAG modulo their direction. This is equivalent to learning the neighbors $N(X_i)$ of each node: its parents and children (see algorithm 1 in [33]). As illustrated in step 3 (learning the parents and the children of each node), the absence of a set of nodes $S_{X_iX_j}$ that separates a particular pair X_i, X_j implies that either $X_i \rightarrow X_j$ or $X_j \rightarrow X_i$. Separating sets are considered in order of increasing size to keep computations as local as possible. Furthermore, if Π_{X_i} and Π_{X_j} are available from steps 1 and 2, the search space can be greatly reduced because $N(X_i) \subseteq \Pi_{X_i}$. On the one hand, let $\mathbb{B}_{X_i}$ be the Markov blanket of variable X_i, if $X_i \notin \mathbb{B}_{X_i}$ by definition X_i is separated from X_j by $S_{X_iX_j} = \Pi_{X_i}$. On the other hand, if $X_j \in \Pi_{X_i}$, most candidate sets can be disregarded because we know that $S_{X_iX_j} \subseteq \mathbb{B}_{X_i} \setminus X_j$ and $S_{X_i,X_j} \subseteq \Pi_{X_j} \setminus X_i$. With the exception of the PC algorithm, which is structured exactly as described in step 3, constraint-based algorithms learn the skeleton by learning each $N(X_i)$ and then enforcing symmetry (step 4: check whether they are symmetric).

Finally, in the third phase, arc directions are established as in [34]. It is important to note that, for some arcs, both directions are equivalent in the sense that they identify equivalent conditional independencies. Therefore, some arcs will be left undirected, and the algorithm will return a completed partially directed acyclic graph identifying an equivalence class containing multiple DAGs. Such a class is uniquely identified by the skeleton learned in steps 3 and 4, and by the v-structures $X_i \rightarrow X_k \leftarrow X_j, X_k \in N \setminus X_j$ learned in step 5 [35]. Additional arc directions are inferred indirectly in step 6 by ruling out those that would introduce additional v-structures (which would have been identified in step 5) or cycles (which are not allowed in DAGs).

2.5.6 Constraint Using Topological Ordering of Nodes

A topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge $X \rightarrow Y$ from vertex X to vertex Y, X precedes Y in the ordering. For instance, the vertices of the graph may represent tasks to be performed, and the edges may represent constraints where one task must be performed before another. In this application, a topological ordering represents a valid sequence for the tasks. A topological ordering is possible if and only if the graph has no directed cycles, meaning it is a directed acyclic graph (DAG). Any DAG has at least one topological ordering, and algorithms are known for constructing a topological ordering of any DAG in linear time. An example of an algorithm that assumes the ordering of nodes to learn the BN structure is K2 [22].

2.6 Previous Results and Related Work

Several machine learning methods have been applied to medical problems. Probably the closest work in the area of acute myocardial infarction (AMI) is the work of [36]. The authors used a residual network with a structure similar to that of a convolutional neural network. This architecture allows a deep neural network to be effectively trained by incorporating skipping connections. The network consisted of a convolutional layer (Conv) and four residual blocks, each of which had two convolutional layers. The output of the last block was connected to a fully connected layer (dense) with a sigmoid activation function. The output of each convolutional layer was rescaled using batch normalisation and passed through a rectified linear activation unit (ReLU).

In another paper [37], the authors compared several over- and under-sampling techniques to deal with the imbalance in the dataset. They compared regularised logistic regression, random forest, boosted gradient machines, and shallow and deep neural networks. A baseline model for comparison was a logistic regression model using a limited set of 'known' risk factors for MI. Hyperparameters were determined using 10-fold cross-validation.

In addition, Kaufmann [38] conducted research to investigate the decision-making process underlying the estimation of cardiac function in patients acutely admitted to the intensive care unit (ICU) based on the current standardised clinical examination using Bayesian methods. Using real data, the study first analysed the probabilistic dependencies between the examiner's estimates and the set of clinically measured variables on which they are based, using a Bayesian network. Second, the accuracy of the cardiac function estimates was assessed by comparing them with cardiac index values measured by critical care ultrasound.

Chapter 3

Main Results

3.1 Heart Attack Mortality Prediction

This part of the thesis was published in [R1,R3], listed in 4.2. Acute myocardial infarction (AMI) is commonly known as a heart attack. A heart attack occurs when an artery leading to the heart becomes completely blocked, and the heart doesn't receive enough blood or oxygen. Without oxygen, cells in that area of the heart die. AMI is responsible for more than half of all deaths in most countries worldwide. Its treatment has a significant socioeconomic impact.

One of the main objectives of our research is to design, analyze, and verify a predictive model of hospital mortality based on clinical data about patients. A model that predicts mortality accurately can be used, for example, to evaluate the quality of medical care in different hospitals. Evaluating hospitals based solely on mortality rates would not be fair to those that frequently treat complicated cases. It seems better to measure the quality of healthcare using the difference between predicted and observed mortality.

A related work was published by [39]. The authors analyze mortality data in U.S. hospitals using the logistic regression model.

3.1.1 Data

Our dataset contains information on 787 patients characterised by 24 variables. Among them, 603 patients are from the Czech Republic [40] and 184 are from Syria. The attributes are listed in Table 3.1. Most of the attributes are real, while four are nominal. Only a subset of attributes were measured for the Syrian patients.

Most records contain missing values, i.e. for most patients only some attribute values are available. Thirty-day mortality is recorded for all patients.

In the Czech Republic, blood test results are reported in millimoles per litre of blood. In Syria, some measurements are reported in milligrams per litre and some in millimoles per litre. We standardise all measurements to millimoles per litre.

Attribute	Code	type	value range in data	Country
Age	AGE	real	[23, 94]	SYR, CZ
Height	HT	real	[145, 205]	CZ
Weight	WT	real	[35, 150]	CZ
Body Mass Index	BMI	real	[16.65, 48.98]	CZ
Gender	SEX	nominal	{male, female}	SYR, CZ
Nationality	NAT	nominal	{Czech, Syrian}	SYR, CZ
STEMI Location	STEMI	nominal	{inferior, anterior, lateral}	SYR, CZ
Hospital	Hospital	nominal	{CZ, SYR1, SYR2}	SYR, CZ
Kalium	K	real	[2.25, 7.07]	CZ
Urea	UR	real	[1.6, 61]	SYR, CZ
Kreatinin	KREA	real	[17, 525]	SYR, CZ
Uric acid	KM	real	[97, 935]	SYR, CZ
Albumin	ALB	real	[16, 60]	SYR, CZ
HDL Cholesterol	HDLC	real	[0.38, 2.92]	SYR, CZ
Cholesterol	CH	real	[1.8, 9.9]	SYR, CZ
Triacylglycerol	TAG	real	[0.31, 11.9]	SYR, CZ
LDL Cholesterol	LDLC	real	[0.261, 7.79]	SYR, CZ
Glucose	GLU	real	[2.77, 25.7]	SYR, CZ
C-reactive protein	CRP	real	[0.3, 359]	SYR, CZ
Cystatin C	CYSC	real	[0.2, 5.22]	SYR, CZ
N-terminal prohormone of brain natriuretic peptide	NTBNP	real	[22.2, 35000]	CZ
Troponin	TRPT	real	[0, 25]	CZ
Glomerular filtration rate (based on MDRD)	GFMD	real	[0.13, 7.31]	CZ
Glomerular filtration rate (based on Cystatin C)	GFCD	real	[0.09, 7.17]	CZ

Table 3.1: Attributes

3.1.2 Preliminary Statistical Analysis

For a preliminary statistical analysis [41], we randomly selected 150 Czech patients and 150 Syrian patients from our dataset, creating two groups of equal size. We considered a subset of the characteristics present in both groups, specifically these variables: age, nationality, sex, STEMI location, and mortality.

As STEMI location is nominal and has three states in most experiments, we transformed it into three binary variables: STEMI.inf, STEMI.ant and STEMI.lat. Nationality is coded as a binary variable where 0 is Czech and 1 is Syrian. Gender is coded as a binary variable where 0 is male and 1 is female. Mortality is also coded as a binary variable where 0 indicates that the patient survived 30 days and 1 indicates that the patient did not survive.

From Figure 3.1, which shows the histogram of age values, we can see that among the patients who didn't survive, a high percentage are young patients from Syria.

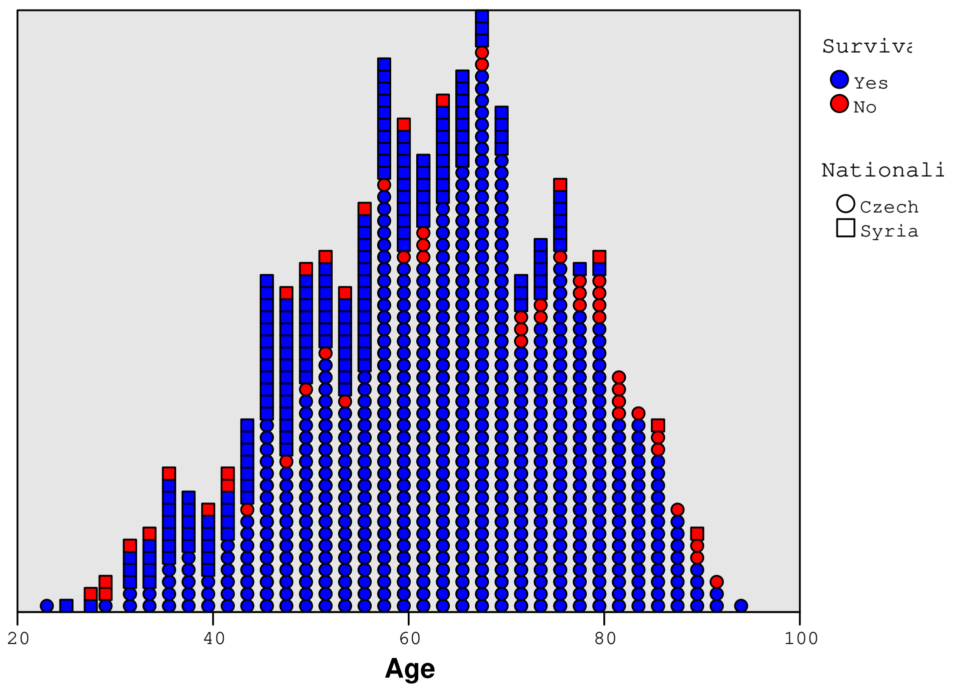


Figure 3.1: Histogram of the age values

Table 3.2: The correlations and their statistical significance

		gender	STEMI loc.	mortality	nationality
age	corr.	0.092	0.001	-0.074	-0.460
	sign.	0.111	0.982	0.199	0.0001
gender	corr.		0.034	0.018	0.133
	sign.		0.557	0.757	0.021
STEMI loc.	corr.			0.104	0.106
	sign.			0.071	0.066
mortality	corr.				0.128
	sign.				0.026

In Table 3.2, we present the correlation matrix, and since it is symmetric, we display only the upper triangular part without the diagonal. Statistically significant correlations (at the 0.05 level) are highlighted.

We can observe a negative correlation between the age of the patients and nationality, where the Czech Republic is encoded as 0 and Syria as 1. Consequently, the average age of Czech patients is greater than that of Syrian patients. There is also a significant difference in the percentage of male and female patients in each country – 28% of patients in the Czech Republic are female, compared to 40.6% in Syria. Additionally, there is a significant difference in mortality rates between Syrian patients (12%) and Czech patients (5.4%).

Table 3.3: The Chi-Square Test of conditional independence

		gender	STEMI loc.	mortality	nationality
age	value	52.63	136.7	102.57	104.78
	sign.	0.821	0.242	0.001	0.001
gender	value		1.605	0.096	5.337
	sign.		0.448	0.756	0.021
STEMI loc.	value			10.678	17.173
	sign.			0.005	0.0001
mortality	value				4.925
	sign.				0.026

The standard chi-square test of conditional independence between two variables reveals (see Table 3.3) that there is a significant dependence between mortality and nationality. There is also a significant dependence between mortality and STEMI location – the patients from Syria with a lateral infarction have the lowest probability to survive.

Table 3.4: The Mann–Whitney U test

		age	gender	STEMI.lat	STEMI.ant	STEMI.inf	nationality
mortality	value	3100	10036	2833	2952	3567	2860
	sign.	0.173	0.757	0.002	0.045	0.748	0.027

We also conducted the Mann–Whitney U test (see Table 3.4) to determine if there is a significant difference in mortality among groups classified by age, gender, STEMI location, and nationality. Based on the test results, we conclude that patients from the Czech Republic have lower mortality rates than Syrians, and patients with lateral infarctions have a lower probability of survival.

Finally, we developed a logistic regression model to describe the relationship between the independent variables and mortality as the dependent variable. The model is defined as follows:

$$\begin{aligned}\text{logit } \mathcal{L}(Y=1|X=x) &= \beta_0 + \beta_1 x_1 + \ldots + \beta_5 x_5 \\ &= -2.375 - 0.006 \cdot x_1 - 0.026 \cdot x_2 + 0.613 \cdot x_3 + 0.916 \cdot x_4 - 0.489 \cdot x_5\end{aligned}$$

where x_1 is the age, x_2 is the gender (0 for male and 1 for female), x_3 is the nationality (0 for Czech and 1 for Syrian), x_4 is the STEMI.lat (0 for no, 1 for yes), and x_5 is the STEMI.ant (0 for no, 1 for yes).

However, only the intercept and the variable STEMI.lat appear to be statistically significant for predicting mortality.

From the preliminary statistical analysis, we can conclude that although the various tests we used do not suggest exactly the same relationships between the studied variables, they mostly agree on a few significant findings:

- In Syria, mortality from AIM is significantly higher than in the Czech Republic—87.3% of Syrian patients survive, while 94.7% of patients from the Czech Republic survive.
- The average age of patients in Syria is lower (with an average difference of 13 years), and there is a higher prevalence of women among patients with AIM in Syria compared to the Czech Republic.
- The STEMI location is related to mortality.

3.1.3 Machine Learning Methods

The preliminary statistical analysis focused on pairwise relations. Since explanatory variables can combine their influence, and the influence of one variable may be mediated by another, it is worthwhile to study the relationships between all variables together. We will do this in two steps: (1) Since mortality prediction is our primary interest, we will compare how different classifiers are able to predict mortality. (2) To gain an overall understanding of the relations between all variables, we will learn a Bayesian network model from the collected data [40].

We will work with different versions of the data, depending on how we treat variables that have more than two states: (1) Real-valued ordinal variables, (2) Discrete-valued variables (with at most five states), and (3) Binary variables. We will discuss the transformation of values in more detail in the following sections.

Our data are incomplete and imbalanced. We will present an idea for dealing with this type of data using tree-augmented naive Bayes (TAN).

3.1.3.1 Ordinal Attributes

In our dataset, we have several categorical variables, often referred to as nominal variables. These are variables with two or more categories. For instance, gender is a categorical variable with two categories: male and female. However, certain machine learning methods require ordinal attributes. These are attributes with values that possess a natural order, quantifying their impact on the class. This ordinality criterion applies even to nominal attributes, such as gender (0 for male, 1 for female) and mortality (0 for survived, 1 for died). In our dataset, most real-valued attributes can be considered ordinal. It's worth noting that there may also be laboratory tests whose values deviate from the normal range in both directions, meaning that both lower and higher values may increase mortality. We will refer to this ordinal data as D.ORD.

3.1.3.2 Discrete Attributes

A discrete variable can take values from a finite set. Certain classification techniques require discrete variables. To obtain statistically reliable estimates of model parameters, it is advisable to minimize the number of values while still capturing significant relationships.

In our case, we have discretized all real-valued attributes. Determining the optimal number and values of splitting points in discretization can be challenging. Fortunately, the Czech National Code Book provides a helpful resource. It classifies numerical laboratory results into nine groups (1, 2, ..., 9) based on age and sex, with Group 5 corresponding to standard values in the standard population. We have further reduced the number of states to 5 by combining some of these groups. We will refer to the data in this discretized form as D.DISCR.

3.1.3.3 Binary Attributes

Binary data consists of variables that can have only two possible states, traditionally denoted as 0 and 1 in accordance with the binary numeral system and Boolean algebra. In our case, all laboratory tests are encoded using two binary attributes. The first attribute takes a value of 0 for standard test values and 1 if the values are decreased. The second attribute takes a value of 0 for standard test values and 1 if the values are increased. The attributes Age, Height, and Weight have been removed. From the demographic group of attributes, only Gender and the Body Mass Index (BMI) have been retained. BMI is encoded using two binary attributes: BMI high (1 if BMI is greater than the mean, 0 otherwise) and BMI low (1 if BMI is less than or equal to the mean, 0 otherwise). We will refer to data in this form as D.BIN.

3.1.3.4 Attribute Selection

Before learning a model, we preprocess the data. Usually, one of the most useful parts of preprocessing is attribute selection, where irrelevant attributes are removed. Attribute selection is a process by which we automatically search for the best subset of attributes in our dataset. The notion of "best" is relative to the problem we are trying to solve, but typically means the highest accuracy. Three key benefits of performing attribute selection on our data are:

- Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.
- Improves Accuracy: Less misleading data means that modeling accuracy improves.
- Reduces Training Time: Less data means that algorithms train faster.

The CfsSubsetEval method of Weka [42] selects subsets of attributes that are highly correlated with the class while having low inter-correlation. We searched the space of all subsets by a greedy best-first search with backtracking. Data D after the application of this attribute selection method will be suffixed as D.AS.

3.1.3.5 Tested Classifiers

For tests, we used a large subset of classifiers implemented in Weka. Classifiers that performed best in the preliminary tests qualified for the final tests. In the final tests, we compared the following classifiers:

- **Decision tree C4.5** [43].
- **Logistic regression** [44].
- **Naive Bayes (NB) classifier** [45], which assumes that the value of a particular explanatory variable (attribute) is independent of the value of any other attribute given the class variable.
- **Bayesian network (BN) classifiers**, including (1) those learned by the K2 algorithm [46] (referred to as BN.K2) and (2) the Tree Augmented Naive Bayes classifier referred to as BN.TAN [12].

  Where all BN algorithms implemented in Weka assume that all variables are discrete finite variables, we will use "NA" in the results of that classification methods for non-discrete data.

  We use the leave-one-out cross-validation as the model evaluation method. It means that N separate times, the classifier is trained on all the data except for one point, and a prediction is made for that point. After that, the average error is computed and used to evaluate the model.

3.1.3.6 Prediction Quality

For each data record classified by a classifier, there are four possible classification results. Either the classifier got a positive example labeled as positive (in our data, the positive example is the patient not survived), or it made a mistake and marked it as negative. Conversely, a negative example may have been mislabeled as a positive one or correctly marked as negative. This defines the following metrics:

- **True Positives (TP):** the number of positive examples labeled as such.
- **False Positives (FP):** the number of negative examples labeled as positive.
- **True Negatives (TN):** the number of negative examples labeled as such.
- **False Negatives (FN):** the number of positive examples labeled as negative.

We used the following measures of prediction quality:

- **Accuracy** measures how often the classifier makes the correct prediction. It is the ratio between the number of correct predictions and the total number of predictions.
$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$
- **Recall**, also known as sensitivity, is the fraction of positive instances that are correctly classified as positive (the rate of true positives).
$$REC = \frac{TP}{TP + FN}$$
- **Precision** is the fraction of true positives over the number of all reported positives.
$$PRE = \frac{TP}{TP + FP}$$
- **F-measure** is the harmonic mean of precision and recall.
$$F = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$
- **Specificity** is the fraction of true negatives over the number of all negatives.
$$SPE = \frac{TN}{FP + TN}$$
- **Area under the ROC curve (AUC)**. The ROC curve shows how the classifier can sacrifice the true positive rate (recall or sensitivity) for the false positive rate (1-specificity) by plotting the TP rate against the FP rate. In other words, it shows you how many correct positive classifications can be gained as you allow for more and more false positives. As an example, in Figure 3.2, we report the ROC curve for the Naive Bayes classifier with the ordinal attributes. Its area under the curve is 0.782.

3.1.3.7 Results of Experiments

In Table 3.5 we compare the results of different classifiers on different versions of data. The C4.5 classifier with D.DISCR has the highest accuracy of 0.942, its recall and precision are also among the best achieved. However, its area under the ROC curve is very low, only 0.371, which suggests that this classifier can not be satisfactorily tuned if we want to sacrifice precision to recall or vice versa.

The contribution of attribute selection method (CfsSubsetEval method of Weka) to the performance of models was pretty good where the accuracy was improved in general except C4.5 with D.ORD, and LOG.REG with D.ORD and D.BIN. Additionally, the AUC and F-measure were improved in most of the models.

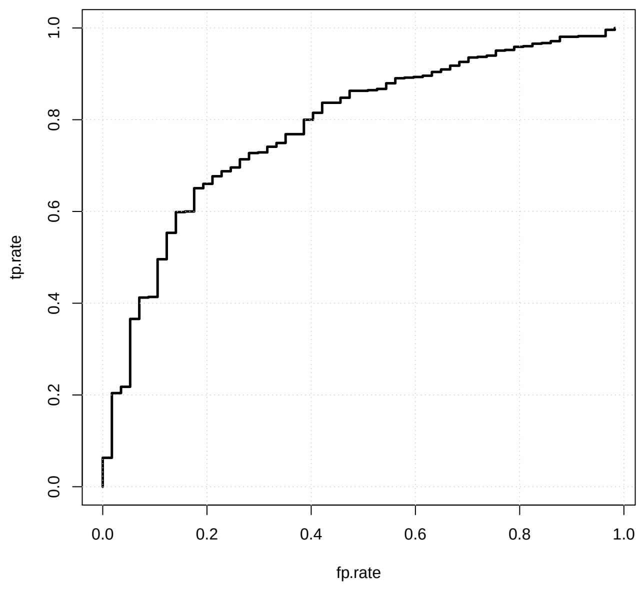


Figure 3.2: ROC for the NB classifier with ordinal attributes

Figure 3.4 shows the tree structure of C4.5 trained on the discrete data. Its structure is surprisingly simple. If the patient is Czech then it is predicted that he will survive, if the patient is Syrian then the LDL cholesterol value should be checked. If it is below 4.78, the patient is predicted to survive, otherwise, if the LDL cholesterol value is between 4.78 and 6.28 then it depends on the Syrian hospital where he/she is being treated. If he/she is treated in the public hospital (SYR1) he/she will die; if he/she is treated in a private hospital (SYR2), he/she will survive. If his/her LDL cholesterol is higher than 6.28, he/she will die (regardless of which Syrian hospital he/she is treated in). The simplicity of the C4.5 classifier is in line with the general recommendation that to avoid overfitting training data models should be as simple as possible. This is probably the best we can learn from the data, but it probably oversimplifies reality. More data would be needed.

The highest accuracy of all the classifiers tested was achieved by Random Forest (RF). The highest area under the ROC curve (AUC) was achieved by the Naive Bayes classifier with ordinal attributes. The highest value of F-measure was achieved by BN.K2 with discrete attributes. by the CfsSubsetEval method of Weka [42]. The learned BN model is actually also a Naive Bayes model – see Figure 3.3. We can conclude that there is no single winner – no classifier that is the best in all the criteria considered.

The classifiers also differ in which variables they consider important for predicting AMI mortality. We believe that learning different classifiers is worthwhile because it

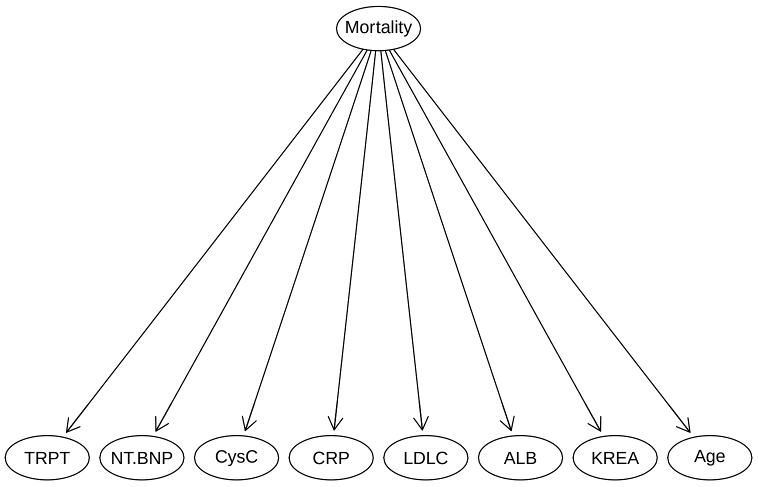


Figure 3.3: BN learned by BN.K2

can help medical professionals to get insight into the problem being modelled.

```
hHospital <= 0: 0 (603.0/36.0)
hHospital > 0
h|   LDLC <= 4.78: 0 (157.86/6.0)
h|   4.78 < LDLC <= 6.28
h|   |   Hospital <= 1: 1 (12.95/2.95)
h|   |   Hospital > 1: 0 (9.0/1.0)
h|   LDLC > 6.28: 1 (4.18/0.18)
```

Figure 3.4: Decision tree C4.5 learned from D.DISCR.

Table 3.5: Results of experiments

Classifier	Criteria	D.ORD	D.ORD.AS	D.DISCR	D.DISCR.AS	D.BIN	D.BIN.AS
Naive Bayes	ACC	0.855	0.925	0.860	0.914	0.875	0.911
	AUC	**0.782**	0.722	0.744	0.781	0.695	0.717
	Recall	0.439	0.158	0.351	0.368	0.246	0.140
	Prec.	0.234	0.450	0.215	0.396	0.203	0.276
	F-measure	0.305	0.234	0.267	0.382	0.222	0.186
C4.5	ACC	0.935	0.933	0.942	0.921	0.926	0.927
	AUC	0.527	0.621	0.371	0.627	0.528	0.273
	Recall	0.263	0.105	0.246	0.123	0.070	0.035
	Prec.	0.625	0.750	0.875	0.368	0.444	0.333
	F-measure	0.370	0.185	0.384	0.184	0.121	0.063
RF	ACC	0.929	0.931	**0.947**	0.934	0.905	0.912
	AUC	0.532	0.559	0.383	0.631	0.529	0.291
	Recall	0.269	0.215	0.244	0.127	0.101	0.095
	Prec.	0.635	0.770	0.891	0.382	0.451	0.337
	F-measure	0.377	0.319	0.385	0.188	0.159	0.132
SVM	ACC	0.932	0.937	0.906	0.923	0.901	0.924
	AUC	0.542	0.629	0.383	0.631	0.529	0.291
	Recall	0.289	0.218	0.239	0.114	0.113	0.085
	Prec.	0.629	0.769	0.889	0.402	0.459	0.341
	F-measure	0.394	0.322	0.377	0.170	0.162	0.133
LOG.REG	ACC	0.930	0.925	0.907	0.919	0.926	0.919
	AUC	0.746	0.755	0.622	0.746	0.675	0.746
	Recall	0.140	0.018	0.193	0.140	0.070	0.140
	Prec.	0.571	0.250	0.289	0.364	0.364	0.364
	F-measure	0.225	0.033	0.232	0.203	0.118	0.203
NB-Tree	ACC	0.932	0.936	0.914	0.920	0.913	0.920
	AUC	0.658	0.480	0.701	0.726	0.701	0.726
	Recall	0.211	0.228	0.228	0.088	0.070	0.088
	Prec.	0.600	0.684	0.310	0.313	0.211	0.313
	F-measure	0.312	0.342	0.263	0.137	0.105	0.137
BN.K2	ACC	NA	NA	0.886	0.918	0.900	0.926
	AUC	NA	NA	0.750	0.775	0.687	0.671
	Recall	NA	NA	0.316	0.368	0.193	0.105
	Prec.	NA	NA	0.265	0.429	0.256	0.462
	F-measure	NA	NA	0.288	**0.396**	0.220	0.171
BN.TAN	ACC	NA	NA	0.908	0.925	0.904	0.927
	AUC	NA	NA	0.721	0.768	0.653	0.642
	Recall	NA	NA	0.193	0.228	0.088	0.053
	Prec.	NA	NA	0.297	0,464	0.179	0.333
	F-measure	NA	NA	0.234	0.306	0.118	0.091

3.2 Learning TAN from Incomplete Data

This part of the thesis was published in [R4], listed in 4.2. Missing data is a common problem that requires consideration in many data mining, machine learning, and pattern recognition applications. Some variables may remain unobservable (i.e., hidden) even for training instances. Nowadays, an increasing number of datasets are available, and most of them are incomplete. Therefore, we aim to find a way to build a new model from an incomplete dataset. Normally, to learn the maximum likelihood Tree-Augmented Naive Bayes (TAN) structure [12], we need complete data, where all instances $(\mathbf{u}_i, i \in \{1, \ldots, n\})$ from $\mathcal{D}$ are complete and do not have any missing values. If the data are incomplete, and an instance has a missing value, we will not use the whole instance in TAN structure learning. In other words, we will not use the other known values from that instance in TAN structure learning. Note that the class is always known, and a missing value in the dataset is denoted by "NA."

Our goal is to learn a Tree-Augmented Naive Bayesian (TAN) model from incomplete data. Some previous work [47] proposes maximizing conditional likelihood for Bayesian Network (BN) parameter learning. The authors apply their method to MCAR (Missing Completely At Random) incomplete data by using available case analysis to find the best TAN classifier. In other work by [48], they also deal with TAN classifiers and the expectation-maximization (EM) principle for partially unlabeled data. In their work, only the variable corresponding to the class can have missing values.

Another work [49] addresses TAN based on the EM principle, where they have proposed an adaptation of the learning process of the Tree-Augmented Naive Bayes classifier from incomplete data. In their work, any variable can have missing values in the dataset. Also, in other work [50], the author uses graphical independence networks (gRain) [51] to compute the class posterior of instances with missing values.

The TAN algorithm can be adapted to learn from incomplete datasets, allowing for the utilization of most available data in TAN structure learning. The procedure is shown in Algorithm 1, where the Conditional Mutual Information (CMI) is defined as:

$$I(X, Y|Z) = \sum_{\mathbf{x},\mathbf{y},\mathbf{z}} f(\mathbf{x}, \mathbf{y}, \mathbf{z}) \log \frac{f(\mathbf{z}) f(\mathbf{x}, \mathbf{y}, \mathbf{z})}{f(\mathbf{x}, \mathbf{z}) f(\mathbf{y}, \mathbf{z})} \tag{3.1}$$

Here, the sum is only over $\mathbf{x}, \mathbf{y}, \mathbf{z}$ such that $f(\mathbf{x}, \mathbf{z}) > 0$ and $f(\mathbf{y}, \mathbf{z}) > 0$, where f is conditional probability distribution function.

Algorithm 1 TAN for Incomplete Data

```
1: procedure CMI($X_i, X_j, C\}$)                ▷ // Conditional Mutual Information
2:     $\overline{\mathcal{D}} = \{\overline{\mathbf{u}}_1, \dots, \overline{\mathbf{u}}_N\}, \overline{\mathbf{u}}_m = (x_i, x_j, c), m \in \{1, \dots, N\}$, such that $\mathbf{u}_m = (x_1, \dots, x_n, c) \in \mathcal{D}$
3:     Foreach $\overline{\mathbf{u}}_m \in \overline{\mathcal{D}}$
4:         If($a_i == NA | a_j == NA$)
5:             Delete $\overline{\mathbf{u}}_m$ from $\overline{\mathcal{D}}$
6:     endfor
7:     Compute $I_p = I(X_i, X_j | C)$ from $\overline{\mathcal{D}}$
8:     return $I_p$
9: Endprocedure
10: Read $\mathcal{D} = \{\mathbf{u}_1, \dots, \mathbf{u}_N\}, \mathbf{u}_m = (x_1, \dots, x_n, c), m \in \{1, \dots, N\}$
11: var:
12: $n$ the number of attribute variables $X$
13: $\mathbb{I}_p[n][n]$ the WeightMatrix;
14: $UG$ the UndirectedGraph;
15: $UT$ the UndirectedTree;
16: $T$ the DirectedTree;
17: TAN the DirectedGraph;
18: Foreach $X_i, i \in \{1, \dots, n-1\}$
19:     Foreach $X_j, j \in \{2, \dots, n\}$
20:         $I_{pij} = CMI(X_i, X_j, C)$
21:         $\mathbb{I}_p[i, j] = I_{pij}$
22:         $\mathbb{I}_p[j, i] = I_{pij}$
23:     EndForeach
24: EndForeach
25: $G$ = ConstructUndirectedGraph($\mathbb{I}_p$)
26: $UT$ = MaximumWeightedSpanningTree($G$);
27: $T$ = MakeDirected($UT$);
28: TAN = AddClass($T$);
```

In Algorithm 1[1], on line 25, we construct a complete undirected graph in which the vertices are the attributes $X_1, \dots, X_n$. Note that the weight of an edge connecting X_i to X_j, where $i \neq j$, is denoted as $I_{p_{ij}} = I(X_i, X_j|C)$. On line 26, we construct a subgraph of G without any cycles and with the maximum possible total edge weight. On line 27, we transform the resulting undirected tree into a directed one by choosing a root variable and setting the direction of all edges outward from it. On line 28, we add the class C to the graph as a node and add edges from C to all other nodes in the graph.

[1]The complexity is $\mathrm{O}(\mathrm{n}^2 \log n)$

Similarly, we can create a procedure in Algorithm 2[2] that allows the Chow-Liu algorithm to handle incomplete data, whereas the standard Chow-Liu algorithm [52] only deals with complete data. The procedure is shown in Algorithm 2, where Mutual Information (MI) is defined as:

$$I(X,Y) = \sum_{\mathbf{x},\mathbf{y}} f(\mathbf{x},\mathbf{y}) \log \frac{f(\mathbf{x},\mathbf{y})}{f(\mathbf{x})f(\mathbf{y})} \tag{3.2}$$

Here, the sum is only over $\mathbf{x},\mathbf{y}$ such that $f(\mathbf{x}) > 0$ and $f(\mathbf{y}) > 0$, where f is conditional probability distribution function. .

The idea behind Algorithms 1, 2 is that we believe if we use more data then the estimates of mutual information and conditional mutual information are more reliable.

3.2.1 Selective Tree Augmented Naive Bayes

The selective tree augmented naive Bayes (STAN) algorithm [53] is a variant of TAN which performs feature subset selection (FSS) [53] and may augment naive Bayes with less than $\mathrm{n} - 1$ arcs by using the joint probability distribution between each variable and the class as a threshold to run FFS. Only the variables which meet a certain level of threshold α are taken into account for the induction of the TAN.

In our experiments, we will select all variables that are not independent from the class, as deemed by the test of independence based on mutual information (MI) with threshold $\alpha = 0.05$. Before learning the augmenting tree(s), STAN filters out variables and inter-feature dependencies (i.e. its G is not necessarily complete). If some direct dependencies between some pair of feature sets are filtered out, also between some variables and the class are filtered out but a maximum spanning tree is obtained over each connected set of features. Each of these trees is then directed as in the case of TAN (see Figure 3.5). The procedure is shown in Algorithm 3[2].

The idea behind Algorithms 3 to select the subsets of attributes that are highly correlated with the class using mutual information.

We will refer to this algorithm as (STAN).

3.2.2 Imbalanced Data

In the case of imbalanced data, classifiers are more sensitive to predicting the majority class and less sensitive to the minority class. Thus, if we don't address this issue, the classification output will be biased, often resulting in always predicting the majority class. Many methods have been proposed in the past few years to deal with imbalanced data. In our research, the mortality rate of patients with myocardial infarction refers to the percentage of patients who have not survived more than 30

[2]The complexity is $\mathrm{O}(\mathrm{n}^2 \log n)$

Algorithm 2 Procedure Chow-Liu for incomplete data

```
 1: procedure MI($X, Y$)                                   ▷ // Mutual Information
 2:     $\overline{\mathcal{D}} = \{\overline{\mathbf{u}}_1, \ldots, \overline{\mathbf{u}}_N\}, \overline{\mathbf{u}}_m = (x, y), m \in \{1, \ldots, N\}, x, y \in \mathbf{u}_m$, such that $\mathbf{u}_m =$
    $(a_1, \ldots, a_n, c) \in \mathcal{D}$
 3:     Foreach $\overline{\mathbf{u}}_m \in \overline{\mathcal{D}}$
 4:         If($x == NA | y == NA$)
 5:             Delete $\overline{\mathbf{u}}_m$ from $\overline{\mathcal{D}}$
 6:     endfor
 7:     Compute $I_p = I(X, Y)$ from $\overline{\mathcal{D}}$
 8:     return $I_p$
 9: Endprocedure
10: $m = n + 1$                                            ▷ //add the Class to WeightMatrix
11: $\mathbb{I}_{p_c}[m][m]$ Chow-Liu WeightMatrix;  ▷ //last row and the last column are for Class C
12: $n$ the number of attribute variables $X$
13: $UT$ the UndirectedTree;
14: $T$ the DirectedTree;
15: Foreach $X_i, i \in \{1, \ldots, n-1\}$
16:     Foreach $X_j, j \in \{2, \ldots, n\}$
17:         $Ic_{p_{ij}} = MI(X_i, X_j)$
18:         $\mathbb{I}_{p_c}[i][j] = I_{p_{ij}}$
19:         $\mathbb{I}_{p_c}[j][i] = I_{p_{ij}}$
20:     EndForeach
21: EndForeach
22: Foreach $A_i, i \in \{1, \ldots, n\}$                 ▷ // MI between Class and each variable
23:     $Ic_{p_{ij}} = MI(A_i, C)$
24:     $\mathbb{I}_{p_c}[i][m] = I_{p_{ij}}$
25:     $\mathbb{I}_{p_c}[m][i] = I_{p_{ij}}$
26: EndForeach
27: $G$ = ConstructUndirectedGraph($\mathbb{I}_{p_c}$)
28: $UT$ = MaximumWeightedSpanningTree($G$);
29: $T$ = MakeDirected($UT$);
```

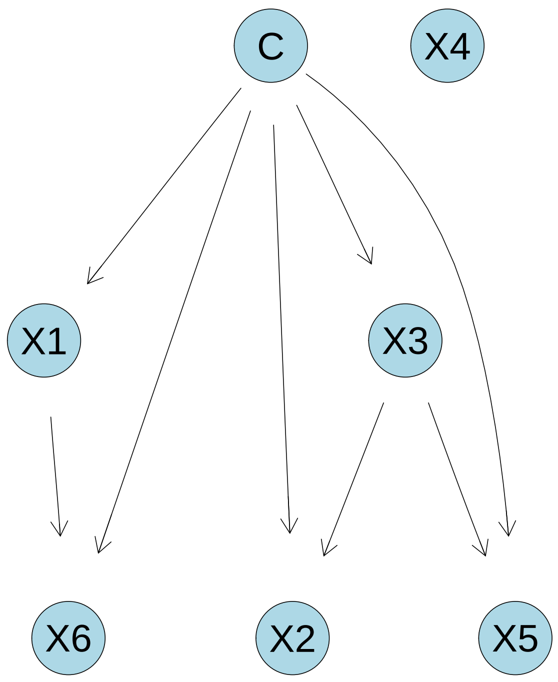


Figure 3.5: STAN

Algorithm 3 Procedure STAN

```
1: $I_C$ the WeightVector between class and variables
2: In Algorithm 1 run lines 1 till 27
3: **Foreach** two connected variables $X_i, X_j, i \neq j$ in $T$
4:     $Ic_{p_ij} = MI(X_i, X_j)$
5:     **If**$(Ic_{p_ij} < 0.05)$
6:         Delete the edge between $X_i$ and $X_j$
7: **EndForeach**
8: **Foreach** $X_i, i \in \{1, \dots, n\}$
9:     $I_{C,X_i} = \mathrm{MI}(X_i, C)$          ▷ // MI is a function in Algoritm 2
10:    **If**$(I_{C,X_i} > 0.05)$
11:        Add edge between the class and $X_i$
12:    **else**
13:        Delete all edges from/to $X_i$
14: **EndForeach**
```

days. The results show that 89% of patients survive, while 11% of patients do not survive, making the data quite imbalanced. One of the most common and simplest strategies for handling imbalanced data is to under-sample the majority class [54,55]. Although different techniques have been proposed in the past, they did not bring any improvement compared to simply selecting samples at random. Therefore, for this analysis, we propose the following steps:

- Let M be the number of samples for the majority class, and N be the number of samples for the minority class, with M being L times greater than N.
- Divide the instances with the majority class into L distinct clusters.
- Train L predictors, where each predictor is trained on only one of the distinct clusters, but on all of the data from the rare class. To clarify, the data from the minority class are used in the training of all L predictors.
- Use model averaging for the L learned predictors as your final predictor. In our case, we will compute conditional mutual information between each pair of attributes $(X_i, X_j), i, j \in \{1, 2, \ldots, n\}, i \neq j$, given the class L times for each pair. Each time we will use only one of the distinct clusters and all data from the minority class, then we will use the average of conditional mutual information for each pair to compute the weight matrix.

We provide a procedure for WeightMatrix computation in Algorithms 1, 2, and 3. We replace its computation with Algorithms 4 that deal with incomplete and imbalanced data.

Algorithm 4 Procedure for WeightMatrix computation with incomplete and imbalance data

```
 1: var
 2:     $M$ number of samples for the majority class
 3:     $N$ number of samples for the minority class
 4:     $\mathcal{D}_T$ instances of the majority class, $\mathcal{D}_T \subset \mathcal{D}$
 5:     $\mathcal{D}_F$ instances of the minority class, $\mathcal{D}_F \subset \mathcal{D}$
 6: integer division $L = M/N$
 7: Divide $\mathcal{D}_T$ to $L$ parts, $\mathcal{D}_{T_k}, k \in \{1, \ldots, L\}$
 8: **Foreach** $k \in \{1, 2, \ldots, L\}$
 9:     $\mathcal{D}_k = \mathcal{D}_{T_k} \cup \mathcal{D}_F$
10:     $\mathbb{I}_{p_k}$ = WeightMatrix for $\mathcal{D}_k$
11: **EndForeach**
12: $\hat{\mathbb{I}}_p$ = the average of $\mathbb{I}_{p_k}, k \in 1, \ldots, L$        $\triangleright$ // $\hat{\mathbb{I}}_p$ is the final WeightMatrix
```

We will refer to TAN, Chow-Liu and STAN which deal with incomplete and imbalanced data as TANI, CLI, and STANI.

3.2.3 Results

To compare the methods, we used the dataset presented in Table 3.1. The results are summarized in Table 3.6. We compared the results of our methods with the following:

- TAN for the incomplete dataset in *bnclassify* [50], which we will refer to as (TB).
- Chow-Liu [52], referred to as (CL).
- EM algorithm [48] for Chow-Liu using *Hugin*[3] [56], which we will refer to as (EMCL).
- Normal TAN [12] after omitting all instances with a missing value.
- SMOTE algorithm [57] for TAN, referred to as (ST).
- Algorithm in [49]. This algorithm deals with TAN based on the EM algorithm, where they have proposed an adaptation of the learning process of the Tree Augmented Naive Bayes classifier from incomplete data, where any variable can have missing values in the dataset. We will refer to it as (FL) on two versions of the dataset.

We use AUC for comparison. Also, we use the 10-fold cross-validation as the model evaluation method. Algorithm TANI with D.BIN has achieved the highest area under the ROC curve (AUC) (0.953) - see Figure 3.6. The results of Algorithm 1 are better than the normal TAN algorithm in both datasets. The SMOTE algorithm with TAN has achieved a better AUC than the AUC of Algorithm 1 with D.DISCR and Algorithm 2 in both datasets.

3.2.4 Quality of Classifiers Tested on Artificial Data

It is difficult to determine which algorithm is better than another one since, in different papers, algorithms are usually tested using different datasets with specific constraints such as dimension, the number and type of attributes, data distribution, the number of classes, missing values, etc.

We decided to use artificial data to enable fair comparisons using the fixed models of TAN. The experiments were repeated ten times on ten different subsets for each MCAR [58] rate on different models. We used datasets generated from the true models summarized in Table A.1 in Appendix A.

3.2.4.1 Results of Experiments

We compare our algorithms Algo1 and STAN with related algorithms FL and TB. We used area under curve (AUC) to compare the algorithms. Figure 3.7 represents the results of all models with all dataset sizes and all MCAR rates.

[3]http://www.hugin.com

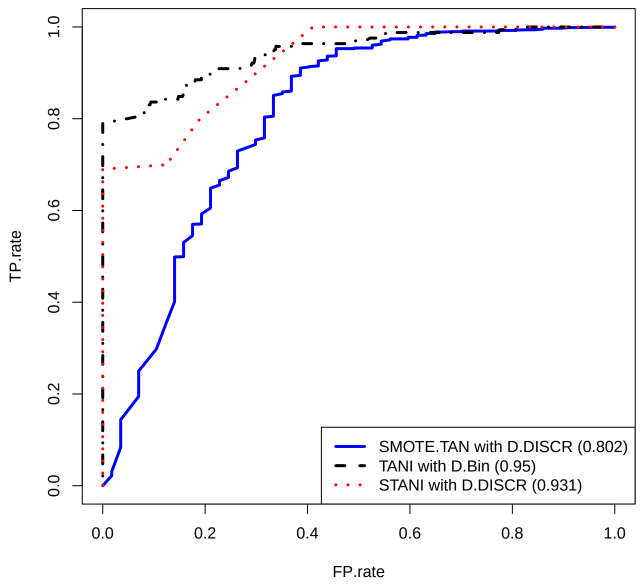


Figure 3.6: AUC

Table 3.6: BN Results

		D.DISCR	D.Bin
TB	AUC	0.804	0.448
FL	AUC	**0.950**	0.871
ST	AUC	0.802	0.818
CL	AUC	0.723	0.690
EMCL	AUC	0.691	0.710
TAN	AUC	0.620	0.670
Algo1	AUC	0.770	0.930
Algo2	AUC	0.750	0.730
STAN	AUC	0.891	0.905
TANI	AUC	0.820	**0.953**
CLI	AUC	0.476	0.895
STANI	AUC	0.931	0.895

We studied the results of T4 model shown in Figure A.2. All results of T4 model - mean (M) of AUC - are shown in Tables 3.7, 3.8, 3.9, 3.10, and 3.11. The results are recorded in the form $(V_{(\pm s)})$ where V is the mean of AUCs and s is the Standard

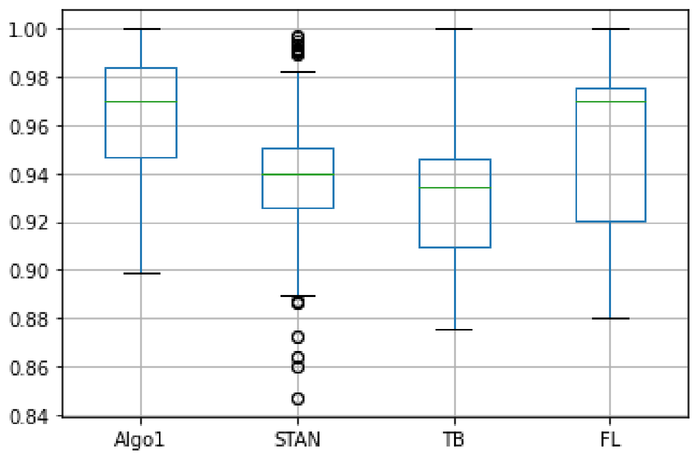


Figure 3.7: The mean (M) of AUC from the resulting models using data generated from all models, summarized in Table A.1 averaged over all data sizes and all MCAR rates.

Deviation (SD).

MCAR	Algo1 ($A1$)	STAN ($A2$)	TB ($A3$)	FL ($A4$)	Wilcoxon tests
10%	$0.972_{(\pm.164)}$	$0.969_{(\pm.122)}$	$0.953_{(\pm.145)}$	$\mathbf{0.979}_{(\pm.168)}$	$A4 >_{0.130} A1 >_{0.220} A2 >_{\mathbf{0.009}} A3$
20%	$0.969_{(\pm.142)}$	$0.938_{(\pm.163)}$	$0.941_{(\pm.104)}$	$\mathbf{0.970}_{(\pm.126)}$	$A4 >_{0.16} A1 >_{\mathbf{0.009}} A3 >_{0.422} A2$
30%	$\mathbf{0.954}_{(\pm.208)}$	$0.903_{(\pm.248)}$	$0.906_{(\pm.158)}$	$0.945_{(\pm.209)}$	$A1 >_{\mathbf{0.006}} A4 >_{\mathbf{0.009}} A3 >_{0.207} A2$
40%	$\mathbf{0.918}_{(\pm.204)}$	$0.886_{(\pm.246)}$	$0.900_{(\pm.165)}$	$0.915_{(\pm.303)}$	$A1 >_{0.210} A4 >_{0.096} A3 >_{\mathbf{0.032}} A2$
50%	$\mathbf{0.894}_{(\pm.204)}$	$0.872_{(\pm.246)}$	$0.865_{(\pm.151)}$	$0.891_{(\pm.316)}$	$A1 >_{0.539} A4 >_{0.052} A2 >_{0.137} A3$

Table 3.7: The mean AUC and its SD over all MCAR and data size 1000

MCAR	Algo1 ($A1$)	STAN ($A2$)	TB ($A3$)	FL ($A4$)	Wilcoxon tests
10%	$0.977_{(\pm.096)}$	$0.971_{(\pm.101)}$	$0.969_{(\pm.138)}$	$\mathbf{0.981}_{(\pm.139)}$	$A4 >_{0.230} A1 >_{\mathbf{0.018}} A2 >_{0.384} A3$
20%	$0.969_{(\pm.122)}$	$0.942_{(\pm.158)}$	$0.940_{(\pm.170)}$	$\mathbf{0.971}_{(\pm.121)}$	$A4 >_{0.452} A1 >_{\mathbf{0.001}} A2 >_{0.361} A3$
30%	$\mathbf{0.959}_{(\pm.194)}$	$0.902_{(\pm.219)}$	$0.911_{(\pm.175)}$	$0.947_{(\pm.208)}$	$A1 >_{\mathbf{0.024}} A4 >_{\mathbf{0.009}} A3 >_{0.080} A2$
40%	$\mathbf{0.920}_{(\pm.205)}$	$0.890_{(\pm.241)}$	$0.900_{(\pm.175)}$	$0.910_{(\pm.225)}$	$A1 >_{\mathbf{0.041}} A4 >_{0.080} A3 >_{0.096} A2$
50%	$\mathbf{0.910}_{(\pm.203)}$	$0.879_{(\pm.252)}$	$0.889_{(\pm.142)}$	$0.902_{(\pm.228)}$	$A1 >_{0.215} A4 >_{\mathbf{0.032}} A3 >_{0.080} A2$

Table 3.8: The mean AUC and its SD over all MCAR and data size 2000

MCAR	Algo1 ($A1$)	STAN ($A2$)	TB ($A3$)	FL ($A4$)	Wilcoxon tests
10%	$0.981_{(\pm.099)}$	$0.974_{(\pm.100)}$	$0.9782_{(\pm.110)}$	$\mathbf{0.982}_{(\pm.131)}$	$A4 >_{0.460} A1 >_{0.340} A3 >_{0.160} A2$
20%	$0.971_{(\pm.110)}$	$0.946_{(\pm.161)}$	$0.952_{(\pm.192)}$	$\mathbf{0.972}_{(\pm.114)}$	$A4 >_{0.570} A1 >_{\mathbf{0.009}} A3 >_{0.187} A2$
30%	$\mathbf{0.963}_{(\pm.194)}$	$0.936_{(\pm.121)}$	$0.921_{(\pm.148)}$	$0.949_{(\pm.149)}$	$A1 >_{\mathbf{0.024}} A4 >_{\mathbf{0.040}} A2 >_{\mathbf{0.009}} A3$
40%	$\mathbf{0.943}_{(\pm.200)}$	$0.931_{(\pm.230)}$	$0.901_{(\pm.176)}$	$0.915_{(\pm.168)}$	$A1 >_{\mathbf{0.004}} A2 >_{0.380} A4 >_{\mathbf{0.001}} A3$
50%	$\mathbf{0.914}_{(\pm.205)}$	$0.886_{(\pm.237)}$	$0.890_{(\pm.160)}$	$0.910_{(\pm.199)}$	$A1 >_{0.380} A4 >_{\mathbf{0.009}} A3 >_{0.312} A2$

Table 3.9: The mean AUC and its SD over all MCAR and data size 5000

MCAR	Algo1 ($A1$)	STAN ($A2$)	TB ($A3$)	FL ($A4$)	Wilcoxon tests
10%	$0.983_{(\pm.095)}$	$0.976_{(\pm.109)}$	$0.979_{(\pm.098)}$	$\mathbf{0.983}_{(\pm.113)}$	$A4 >_{0.460} A1 >_{0.340} A3 >_{0.215} A2$
20%	$0.973_{(\pm.102)}$	$0.951_{(\pm.147)}$	$0.958_{(\pm.100)}$	$\mathbf{0.974}_{(\pm.115)}$	$A4 >_{0.784} A1 >_{\mathbf{0.004}} A3 >_{0.116} A2$
30%	$\mathbf{0.966}_{(\pm.193)}$	$0.938_{(\pm.124)}$	$0.924_{(\pm.130)}$	$0.953_{(193)}$	$A1 >_{\mathbf{0.018}} A4 >_{\mathbf{0.009}} A2 >_{\mathbf{0.009}} A3$
40%	$\mathbf{0.945}_{(\pm.178)}$	$0.923_{(\pm.185)}$	$0.903_{(\pm.172)}$	$0.919_{(\pm.171)}$	$A1 >_{\mathbf{0.002}} A2 >_{0.187} A4 >_{\mathbf{0.003}} A3$
50%	$\mathbf{0.921}_{(\pm.202)}$	$0.900_{(\pm.203)}$	$0.891_{(\pm.162)}$	$0.915_{(\pm.167)}$	$A1 >_{0.246} A4 >_{\mathbf{0.024}} A2 >_{0.116} A3$

Table 3.10: The mean AUC and its SD over all MCAR and data size 7000

MCAR	Algo1 ($A1$)	STAN ($A2$)	TB ($A3$)	FL ($A4$)	Wilcoxon tests
10%	$0.982_{(\pm.099)}$	$0.979_{(\pm.097)}$	$0.980_{(\pm.085)}$	$\mathbf{0.985}_{(\pm.089)}$	$A4 >_{0.539} A1 >_{0.116} A3 >_{0.340} A2$
20%	$0.976_{(\pm.073)}$	$0.956_{(\pm.141)}$	$0.969_{(\pm.103)}$	$\mathbf{0.976}_{(\pm.127)}$	$A4 >_{0.830} A1 >_{0.065} A3 >_{0.052} A2$
30%	$\mathbf{0.968}_{(\pm.177)}$	$0.943_{(\pm.112)}$	$0.948_{(\pm.140)}$	$0.957_{(\pm.191)}$	$A1 >_{\mathbf{0.024}} A4 >_{\mathbf{0.032}} A3 >_{0.570} A2$
40%	$\mathbf{0.948}_{(\pm.194)}$	$0.932_{(\pm.157)}$	$0.910_{(\pm.179)}$	$0.921_{(\pm.170)}$	$A1 >_{\mathbf{0.002}} A2 >_{\mathbf{0.006}} A4 >_{\mathbf{0.001}} A3$
50%	$\mathbf{0.933}_{(\pm.194)}$	$0.906_{(\pm.173)}$	$0.909_{(\pm.173)}$	$0.921_{(\pm.161)}$	$A1 >_{0.137} A4 >_{\mathbf{0.004}} A3 >_{0.380} A2$

Table 3.11: The mean AUC and its SD over all MCAR and data size 10000

Algo1 algorithm achieved best results in all datasets where MCAR rates are higher than 20%. In the other data-sets the FL achieved the best results. Also, TB algorithm achieved the worst results in all data-sets and all MCAR rates. Also from Figures 3.8, and 3.9, we can see that FL was better than Algo1 in the lowest MCAR rate (20%,10%) but not in the highest MCAR rate in all datasets.

Note that standard deviation of results are quite high compared to the means and the confidence intervals overlap quite a lot. However, since the distributions of AUC cannot be assumed to be normally distributed it is not possible to derive directly from these values an (intuitively correct) conclusion that the results of the methods are not statistically significantly different. For that, a non-parametric test that can be used to compare two related samples to assess whether their mean ranks differ. Appropriate seems to be the Wilcoxon signed-rank test in our case. In other words. We did the test for algorithms in each row in each table independently. This allows to see in which situations algorithms statistically differ. We reported the results of the Wilcoxon test in the form of an additional column attached to each table (Tables 3.7, 3.8, 3.9, 3.10, and 3.11), and containing an expression in the form $(A_i >_V A_j >_{V'} A_k >_{V''} A_m)$. Where the values of V, V', V'' are p-values of the Wilcoxon tests between the algorithms with neighboring values of means. The algorithms are ordered $(A_i, A_j, A_k, A_m\ )$ according to their mean AUC values. For simplicity, we report in the tables p-value only for tests with neighboring mean AUC values. The relations are not necessarily transitive but, in our opinion, this suffices to show methods dominance.

From Wilcoxon test results we can see that the FL algorithm in all data_sets with the MCAR rates 10% and 20% achieved p-value higher than the significance level alpha $=$ 0.05 with Algo1. So, we cannot conclude that FL is significantly better than Algo1, but they are significantly better than the TB algorithm. Algo1 in data_sets size higher than 1000 with the MCAR rates (30% and 40%)) achieved p-value less than the significance level alpha $= 0.05$ with its neighbor which means it is significantly better. Finally, Algo1 in all data_sets with the MCAR rate 50% and in data_sets size 1000 with the MCAR rate 40% achieved p-value higher than the significance level alpha $= 0.05$ which means it's not significantly better.
The STAN algorithm performs significantly better than TB in data_sets size 1000 with the MCAR rate 10% and in data_sets size 5000 and 7000 with the MCAR rate 30% (the p-values are less than the significance level alpha $= 0.05$). Also, TB is significantly better than STAN in the data size 1000 with the MCAR rate 40% .

Table 3.12 shows the results of TAN and STAN algorithms using complete data-set. The TAN algorithm is significantly better than STAN in all data-sets.

Figures 3.10, 3.11, 3.12, and 3.13 show the behavior of algorithms for all data-sets in all MCAR rates. The results are directly proportional to the size of the data and inversely proportional to the MCAR rate.

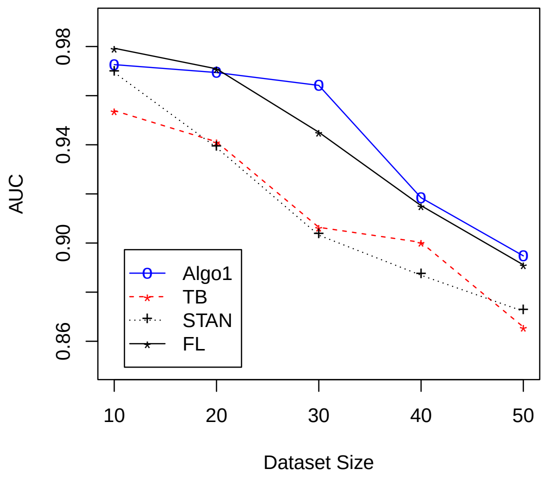


Figure 3.8: The AUC with data size 1000

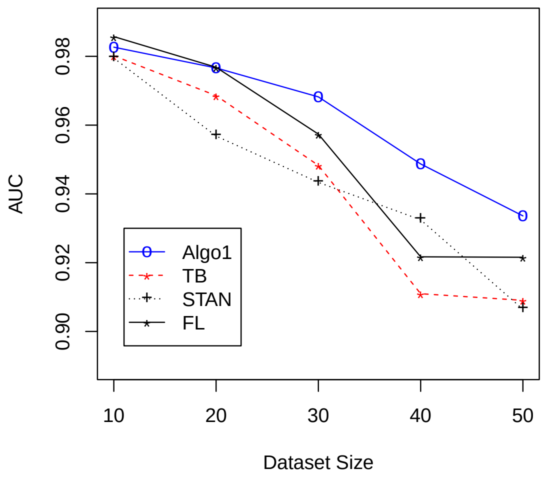


Figure 3.9: The AUC with data size 10000

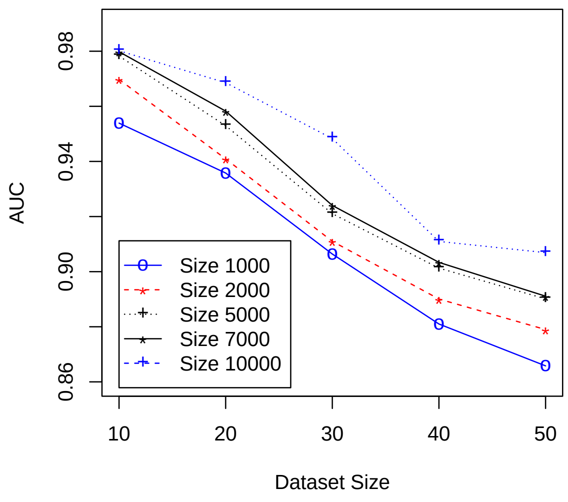


Figure 3.10: The TB Algorithm

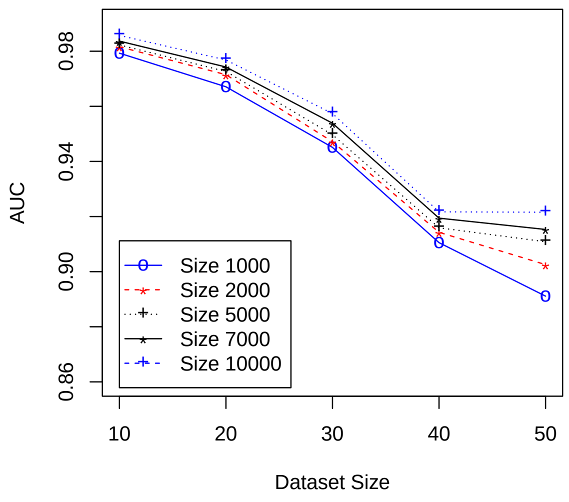


Figure 3.11: The FL Algorithm

Data_Size	1000	2000	5000	7000	10000
TAN	0.978	0.980	0.982	0.984	0.987
STAN	0.966	0.971	0.974	0.979	0.981
Wilcoxon tests	0.01	0.009	0.007	0.005	0.006

Table 3.12: Comparing results of STAN and TAN using complete data

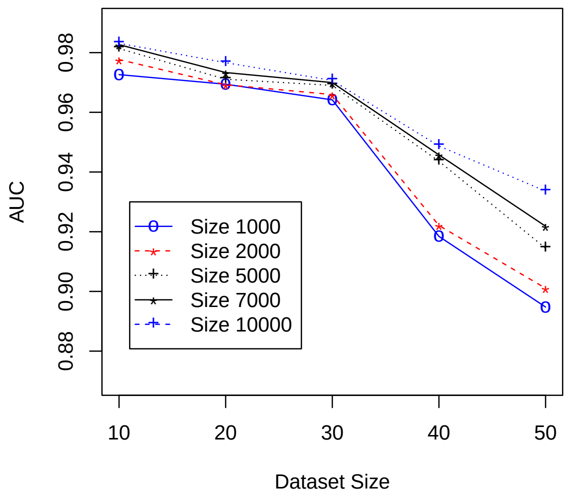


Figure 3.12: The Algo1 Algorithm

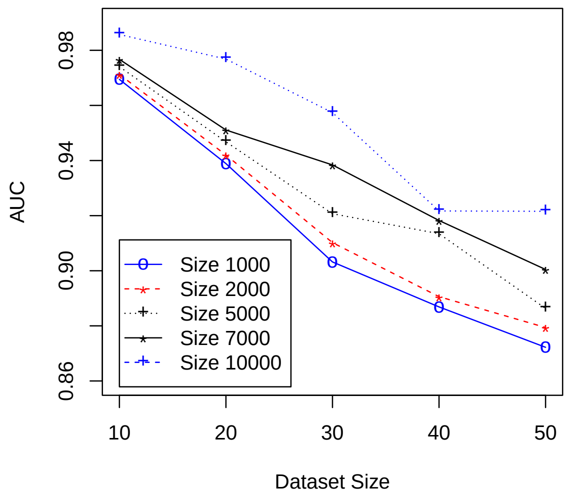


Figure 3.13: The STAN Algorithm

3.2.4.2 TAN Structure Learned

It's important to compare the TAN structures with the real ones to learn where the threshold of the sub-dataset size is for learning the correct TAN structure. From our experiments, we found that we were not able to learn the correct TAN structures when the complete sub-dataset was less than 800. In addition, we couldn't build the correct TAN structure from subsets with a size of 1000 and an MCAR rate higher than 20% for the Algo1 and FL algorithms, but with an MCAR rate higher than 10% for the TB algorithm.

To compare the graphs, we used a numerical measure called the F1 score, which combines recall (the percentage of edges in the original model that are also in the learned model) and precision (the percentage of edges in the learned model that are not in the original model) into a single metric by taking their harmonic mean.

Figure 3.14 shows how the algorithms behave when learning the structures. We observe that FL has a better F1 score and Algo1 is better than TB.

Figure 3.16 (referred to as Algo1_40) shows the structure learned by Algo1 from a

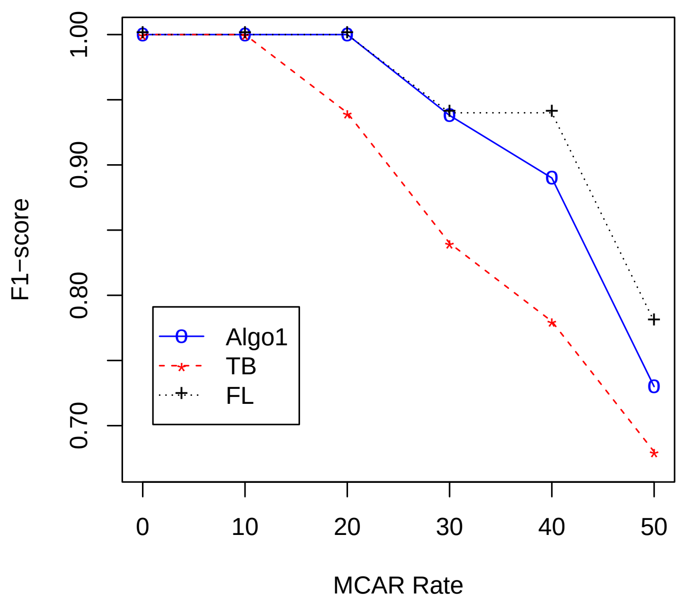


Figure 3.14: F1-score compare essential graphs

dataset with a size of 1000 and a 40% MCAR rate. The arcs between C1 and C4 do not match; the true network contains C1 - C4, while the learned network contains no arc between C1 and C4. Similarly, the arcs between C3 and C4 do not match; the true network contains no arc between C3 and C4, but the learned network contains C3 - C4. The recall for this structure is 0.894.

In Figure 3.15 (referred to as Algo1_50), which shows the structure learned by Algo1 from a dataset with a size of 1000 and a 50% MCAR rate, there are discrepancies in the arcs. Specifically, there are no matching arcs between C1 and C2; the learned network contains no arc between C1 and C2, while the true network contains C1 - C2. Additionally, the learned network contains a C1 - C6 edge that does not exist in the true network. The recall for this structure is 0.7368.

Figure 3.17 (referred to as FL_50) displays the FL structure from a dataset with a size of 1000 and a 50% MCAR rate. Several arcs do not match with the true network. Notably, there are arcs between C1 and C3, but the true network contains no arc between C1 and C3. Similarly, the arcs between C1 and C7 do not match; the learned network contains C1 - C7, while the true network contains no such arc. There are also disparities between C2 and C6, C3 and C4, C3 and C7, C4 and C9, C5 and C6, and C8 and C9. Table 3.13 summarizes these differences, using a checkmark (✓) to indicate existing edges and a multiplication symbol ($\times$) to indicate missing edges.

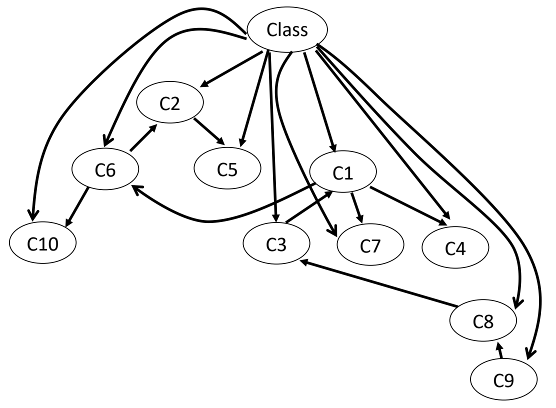


Figure 3.15: Algo1 sub-set_1000 MCAR rate 50%

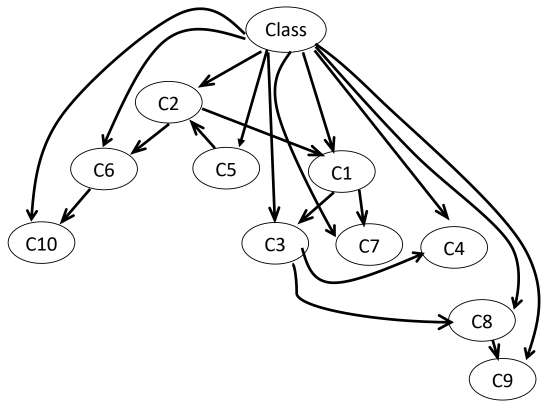


Figure 3.16: Algo1 sub-set_1000 MCAR rate 40%

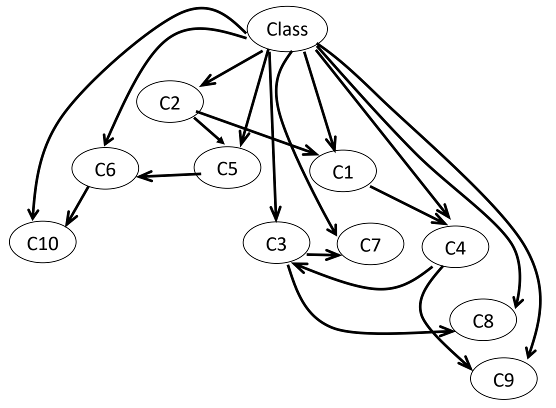


Figure 3.17: FL sub-set_1000 MCAR rate 50%

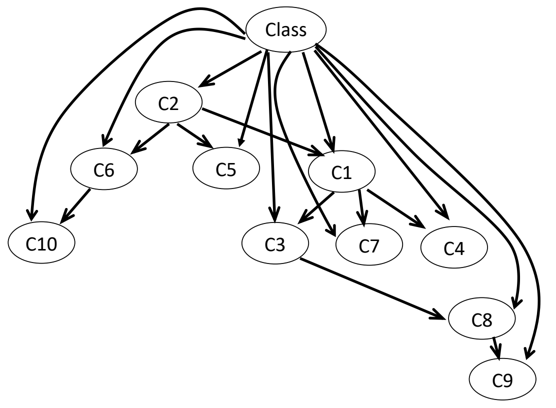


Figure 3.18: True TAN

Table 3.13: Algorithms Structure Comparing

Edage	Algo1_50	Algo1_40	FL_50	True TAN
C1 _ C6	✓			×
C2 _ C1	×			✓
C3 _ C4		✓		×
C1 _ C4		×		✓
C2 _ C6			×	✓
C4 _ C9			✓	×
C1 _ C3			×	✓
C1 _ C7			×	✓
C5 _ C6			✓	×
C3 _ C7			✓	×
C3 _ C4			✓	×
C8 _ C9			×	✓

3.3 Learning the Structure of BNs

This part of the thesis was published in [R5], listed in 4.2. Following up the previous results, we provide a new approach to learning optimal Bayesian network (BN) structures from incomplete data based on the BIC score function using a mixture model to handle missing values. We have compared the proposed approach with other methods. Our experiments have been conducted on different models, some of them Belief Noisy-Or (BNO) ones. We have performed experiments using datasets with values missing completely at random having different missingness rates and data sizes. We have analyzed the significance of differences between the algorithm performance levels using the Wilcoxon test. The new approach typically learns additional edges in the case of Belief Noisy-or models. We have analyzed this issue using the Chi-square test of independence between the variables in the true models; this approach reveals that additional edges can be explained by strong dependence in generated data. An important property of our new method for learning BNs from incomplete data is that it can learn not only optimal general BNs but also specific Belief Noisy-Or models.

3.3.1 Structural Learning with Pruning

Statistical testing is a method of reducing the set of potential DAGs. Another approach to reducing this set is to use constraints provided by experts. Besides that, we can use structural constraints similar to in [30]. The structural constraints can be applied locally as long as they include only one node and its parents.

Algorithm 5 represents an approach to learning the optimal structure of a BN using the constraint rules and a decomposable score [10]. The main function of the algorithm is to compute a collection of candidate parent sets for each variable. Then we optimize across this collection by selecting one parent set for each variable, without creating directed cycles while maximizing the total score. The following corollary can be used to reduce the numbers of the collections for candidate parents. It represents a special case of Lemma 1 in [59].

Lemma 3.3.1. *Let X_i be a variable and $\Pi^{'}$ be a candidate parent set for X_i. Suppose that $BIC(X_i|\Pi^{'}) < BIC(X_i|\{\})$. Then $\Pi^{'}$ can be safely ignored from the candidate parent sets.*

Proof. The proof uses the decomposability of the BIC score. Let $G^{'}$ and G be DAGs that differ only on the parent set of X_i where $\Pi^{'}$ is the parent set of X_i in $G^{'}$ and Π is the parent set of X_i in G. Suppose $\Pi \subset \Pi^{'}$. Therefore, if $G^{'}$ does not contain directed cycles then G cannot contain them either. This fact, together with $BIC(X_i|\Pi) > BIC(X_i|\Pi^{'})$, implies thast $G^{'}$ is not BIC optimal. This statement also holds if the candidate subset is the empty set $\Pi = \{\}$.

□

Let us also note that, if a dataset is generated from a BN having the empty graph as its structure and this dataset is large enough then, for any parent set $\{\Pi_{X_i} \neq \phi\}$, it holds that $BIC(X_i|\{\}) > BIC(X_i|\Pi_{X_i})$. This implies that the variables are independent and the penalty for larger parent sets makes the BIC value worse for all nonempty parent sets.

Algorithm 5 Parent sets evaluation for the BN structure learning algorithm

```
1:  **Input:**
2:      $\mathcal{D}$: a data set
3:      $m$: an integer representing the number of variables in $\mathcal{D}$
4:  **Output:** Accepted sets of parents for each node
5:  **Phase 1: initialize the parameters**
6:      $g_i = (V, E)$
7:      $S_i$: BIC score of $g_i$
8:      $\mathbb{Q}_i$: priority queue of triples $(X_i, \Pi_{X_i}, S_i\ )$ ordered by $S_i$
9:  **Phase 2**: mscour$(X_i, \mathcal{D})$                ▷ function to find the min(BIC) score
10:     $S_i$ = the BIC score of $g_i$ where only $X_i$ is included
11:     **return** $S_i$
12: **Phase 3: find the accepted $\mathbb{Q}_i$ for $X_i$**
13:     $\mathbb{Q}_i$ is empty
14:     $S^* =$ mscour$(X_i, \mathcal{D})$
15:     $\Pi_{X_i}$ is a parent set for $X_i$
16:     add $(X_i, \Pi_{X_i}, S^*)$ to $\mathbb{Q}_i$
17:     For each $X_k, k \in \{1, \ldots, m\}$ do:
18:         add $X_k$ to $\Pi_{X_i}$
19:         $S_{ki}$ = the BIC score of the updated $\Pi_{X_i}$
20:         if$(S_{ki} > S^*)$
21:             add $(X_i, \Pi_{X_i}, S_{ki})$ to $\mathbb{Q}_k$
22:             For each $X_j, j \in \{1, \ldots, m\}, i \neq j \neq k$, do:
23:                 add $X_j$ to $\Pi_{X_i}$
24:                 $S_{ki}$ = the BIC score of the updated $\Pi_{X_i}$
25:                 if$(S_{ki} > S^*)$
26:                     add $(X_i, \Pi_{X_i}, S_{ki})$ to $\mathbb{Q}_k$
27:                 else delete $X_j$ from $\Pi_{X_i}$
28:             end for
29:         else delete $X_k$ from $\Pi_{X_i}$
30:     end for
31:
```

The $g_i = (V, E)$ in Phase 1 from Algorithm 5 is a DAG containing the set of nodes $V = \{X_i, \Pi_{X_i}\}$ and the set of arcs $E = \{(X_o, X_i), \forall X_o \in \Pi_{X_i}\}$, in another world it is a DAG containing a node and its candidate parent set. Algorithm 5 considers all possible parent sets that can lead to an optimal BN. Its implementation is based on [10]. After Phase 3, we find a DAG with the highest BIC from among the variables given the candidate parent sets of each variable. That is done using GOBNILP [23] tool[4] which is a smart algorithm using integer linear programming. We will refer to this algorithm as A1.

One of the axioms of the pruning rules stated in the literature states that if a candidate subset has a better score than another candidate set and the first candidate set is a subset of the second candidate set, it is safe to disregard that second candidate set due to the decomposability of score functions. We have applied the pruning rule as formalized in the Lemma 3.3.1 in Algorithm 5. That algorithm will reduce the collection of accepted parent sets for each node by discarding all parent sets which do not meet the criteria.

3.3.2 Incomplete Datasets

One of the widespread problems in data mining and machine learning is incomplete data. Values may be missing even from training instances. Nowadays more and more datasets are available, but most of them are incomplete. Therefore, machine learning must cope with this problem. Normally, to learn the BN structure using A1 algorithm [30], we need complete data, such that all instances $\mathbf{u}_i \in \mathcal{D}, \quad i \in \{1, \ldots, n\}$ are complete and don't have any missing values. In the case of incomplete data and an instance which has a missing value, A1 does not use this instance in the BN structure learning.

3.3.2.1 Product Distribution Mixtures to Handle Incomplete Data

Because of incomplete data, most methods in machine learning cannot be applied. An easy way to deal with this problem is completing the data by simply omitting the incomplete vectors or removing the incomplete variables. But this approach has a weakness: we may lose a massive part of the available information. Another alternative is to use an estimation to replace the missing values [60] (i.e., put in estimates of the missing values). However, for certain reasons, the estimated values have to be typical, and the natural variability of the data will be partially restricted. For that, the product mixture model gives us a better way to directly apply the EM algorithm to complete the dataset [11]. We will refer to this approach as EM-Mixture.

Considering finite mixtures we assume that:

[4] https://www.cs.york.ac.uk/aig/sw/gobnilp/

$$\begin{aligned} P(X) &= \sum_{j=1}^{r} w_j F(X = x|j), && (3.3) \\ F(X = x|j) &= \prod_{i=1}^{m} f(x_i|j). && (3.4) \\ \sum_{j=1}^{r} w_j &= 1 && (3.5) \end{aligned}$$

where $w_j > 0$ is a probabilistic weight of the j-th mixture component, F is the joint probability distribution, f is the conditional probability distribution of the variable $X_i, i \in 1, \ldots, m$, and r is the number of components. Note that the product components do not imply that the involved variables are independent. In this sense, the mixture model (3.3) is not restrictive [61]. It is easy to verify that, by increasing the number of components r, we can describe any discrete probability distribution in the form (3.3).

To estimate the mixture parameters, we maximize the log-likelihood function:

$$LL(\theta) = \sum_{k=1}^{n} \log P(\mathbf{u}^{(k)})$$

where n is the number of records in the dataset $\mathcal{D}$ and $\mathbf{u}^{(k)}$ is the k-th datavector from $\mathcal{D}$. We will use the EM algorithm to maximize the log-likelihood function.

Next, we explain how the learned product mixture model will be used to fill in the missing values. Let $\mathbf{C} = \{i_1, i_2, \ldots, i_k\}$ be a subset of $\mathbf{M} = \{1, 2, \ldots, m\}$ such that the corresponding sub-vector

$$\mathbf{u_C} = (x_{i_1}, x_{i_2}, \ldots, x_{i_k})$$

is complete. Then, under the product mixture model, we can compute the marginal probability of $\mathbf{u_C}$ as

$$\begin{aligned} P_{\mathbf{C}}(\mathbf{u_C}) &= \sum_{j=1}^{r} w_j F_C(\mathbf{u_c}|j) && (3.6) \\ F_{\mathbf{C}}(\mathbf{u_C}|j) &= \prod_{i \in \mathbf{C}} f(x_i|j) \ . && (3.7) \end{aligned}$$

Let z be an index of a variable unobserved in $\mathbf{u}$, i.e., $z \in \mathbf{M} \setminus \mathbf{C}$. Under the product mixture model, we can compute the conditional distribution of the missing value $\mathbf{u}_z$ given the complete part $\mathbf{u_C}$ with $P_{\mathbf{C}}(\mathbf{u_C}) > 0$ as

$$\begin{aligned} P_{z|\mathbf{C}}(\mathbf{u}_z|\mathbf{u_C}) &= \frac{P_{z,\mathbf{C}}(\mathbf{u}_z, \mathbf{u_C})}{P_{\mathbf{C}}(\mathbf{u_C})} \\ &= \sum_{j=1}^{r} W_j(\mathbf{u_C}) F_z(\mathbf{u}_z|j) \end{aligned}$$

where $W_j(\mathbf{u_C})$ are the conditional component weights:

$$W_j(\mathbf{u_C}) = \frac{w_j F_{\mathbf{C}}(\mathbf{u_C}|j)}{\sum_{j=1}^{r} w_j F_{\mathbf{C}}(\mathbf{u_C}|j)} \enspace .$$

We thus compute the probability distribution $P_{z|\mathbf{C}}(\mathbf{u}_z|\mathbf{u_C})$ for each missing value of each data vector $\mathbf{u} \in \mathcal{D}$ with a missing value. There are several ways of using this probability distribution to fill in the missing value of X_z in $\mathbf{u}$ – in this paper, we select value $\mathbf{u}_z$ maximizing $P_{z|\mathbf{C}}(\mathbf{u}_z|\mathbf{u_C})$ over all values of X_z.

The last step of our presentation is the description of adapting the EM algorithm for learning product mixture models such that it is applicable to incomplete data. Given a data vector $\mathbf{u} \in \mathcal{D}$ and a variable X_i with index $i \in \{1, 2, \ldots, n\}$, let $\mathcal{N}(\mathbf{u})$ be the subset of indices of the available variables (i.e., observed in that data) of $\mathbf{u}$, and $\mathcal{D}(i) \subset \mathcal{D}$ be the subset of vectors with observed values of variable X_i:

$$\begin{aligned} \mathcal{N}(\mathbf{u}) &= \{v \in \{1, 2, \ldots, n\} : \mathbf{u}_v \text{ observed in } \mathbf{u}\} \\ \mathcal{D}(i) &= \{\mathbf{u} \in \mathcal{D} : i \in \mathcal{N}(\mathbf{u})\} \end{aligned}$$

In Algorithm 6, we present the modification of the EM algorithm for the product mixture model for incomplete data. For $x_v \in \mathcal{X}_v$, $v \in \{1, 2, \ldots, n\}$, and $j = 1, \ldots, r$, we use $f(x_v|j)$ to denote the conditional probability of observing value x_v of variable X_v given the component j. The initialization of the EM-Mixture algorithm (presented in Algorithm 6) is performed using the partitions obtained from agglomerative hierarchical clustering implemented in the function *hc* of the R package *mclust* [62]. In our algorithm, the symbol $\delta(x, y)$ denotes the standard delta function equal to one if $x = y$ and equal to zero otherwise.

Algorithm 6 EM-Mixture

```
1:  Input:
2:      $\mathcal{D}$ is a data set
3:  Output: a completed data set
4:  Phase 1: initializing:
5:    $w_j,\ j = 1, \dots, r$
6:
7:    $F_v(x_v|j)$,  for $x_v \in \mathcal{X}_v$, $v \in \{1, 2, \dots, n\}$, and $j = 1, \dots, r$
8:     $L = -\infty$
9:  Phase 2: modified EM
10:   repeat
11:     E-Step:
```

$$\begin{aligned} q(j|\mathbf{u}) &= \frac{w_j \prod_{v \in \mathcal{N}_{(\mathbf{u})}} F_v(\mathbf{u}_v|j)}{\sum_{l=1}^{r} w_l \prod_{v \in \mathcal{N}_{(\mathbf{u})}} F_v(\mathbf{u}_v|l)}, \quad \text{for } \mathbf{u} \in \mathcal{D},\ j = 1, \dots, r \\ w_j &= \frac{1}{|\mathcal{D}|} \sum_{\mathbf{u} \in \mathcal{D}} q(j|\mathbf{u}), \quad \text{for } j = 1, \dots, r \end{aligned}$$

M-Step: for $x_v \in \mathcal{X}_v$, $v \in \{1, 2, \dots, n\}$, and $j = 1, \dots, r$

$$\begin{aligned} F_v(x_v|j) &= \frac{\sum_{\mathbf{u} \in \mathcal{D}(v)} \delta(x_v, \mathbf{u}_v) \cdot q(j|\mathbf{u})}{\sum_{\mathbf{u} \in \mathcal{D}(v)} q(j|\mathbf{u})} \\ L' &= \sum_{\mathbf{u} \in \mathcal{D}} \log \left[\sum_{j=1}^{r} w_j \prod_{v \in \mathcal{N}(\mathbf{u})} F_v(\mathbf{u}_v|j) \right] \\ \mathcal{Q} &= L' - L \\ L &= L' \end{aligned}$$

```
      until $\mathcal{Q} \le \varepsilon$
12:
```

The EM algorithm converges monotonically to a local or global maximum or a saddle point of the log-likelihood function L in the sense that the sequence of $\{L^t\}_{t=0}^{\infty}$ does not decrease. The presence of a local maximum makes the starting point of the procedure influential; hence it is selected at random. We use the value of $\varepsilon = 0.005$ to terminate the main loop of the algorithm. The sequence of log-likelihood values generated by E-Step and M-Step is non-decreasing [11] (i.e., $L^{'} \geqslant L$).

Theorem 3.3.2. *[11]. The sequence of log-likelihood values generated by E-Step and M-Step is non-decreasing (i.e., $L^{'} \geqslant L$).*

Proof. Since Kullback–Leibler follows that information divergence is positive for any two discrete probability distributions [63].

$$\frac{1}{|\mathbf{U}|}\sum_{X_{=x}\in\mathbf{U}}\sum_{m\in\mathcal{M}} q(m|X)\left[log\frac{q(m|X)}{q^{'}(m|X)}\right] \geqslant 0$$

Substitution from E-Step:

$$\frac{1}{|\mathbf{U}|}\sum_{X\in\mathbf{U}}\sum_{m\in\mathcal{M}} q(m|X)\left[log\ \frac{P^{'}(X)}{P(X)} + log\ \frac{w_m F(X{=}\ x|m)}{w^{'}_m F^{'}(X = x|m)}\right] \geqslant 0$$

The first term as we Know that it is equal to the increment of the criterion L:

$$\begin{aligned} L^{'} - L &\geqslant \frac{1}{|\mathbf{U}|}\sum_{X\in\mathbf{U}}\sum_{m\in\mathcal{M}} q(m|X)\left[log\ \frac{w^{'}_m F^{'}(X = x|m)}{w_m F(X = x|m)}\right] \\ &\geqslant \sum_{m\in\mathcal{M}}\left[\frac{1}{|\mathbf{U}|}\sum_{X\in\mathbf{U}} q(m|X)\right] log\ \frac{w^{'}_m}{w_m} + \sum_{m\in\mathcal{M}}\frac{1}{|\mathbf{U}|}\sum_{X\in\mathbf{U}} q(m|X) log\ \frac{F^{'}(X{=}\ x|m)}{F(X{=}\ x|m)} \end{aligned}$$

By using substitution from M-Step we obtain that:

$$L^{'} - L \geqslant \sum_{m\in\mathcal{M}} w^{'}_m log\ \frac{w^{'}_m}{w_m} + \sum_{m\in\mathcal{M}}\frac{1}{|\mathbf{U}|}\sum_{X\in\mathbf{U}} q(m|X) log\ \frac{F^{'}(X_{=x}|m)}{F(X_{=x}|m)}$$

Note that

$$\sum_{m\in\mathcal{M}} w^{'}_m log\ \frac{w^{'}_m}{w_m} \geqslant 0$$

According to M-Step definition, we find that the function $F^{'}(.|m)$ maximizes the right-hand side, i.e.:

$$\sum_{m\in\mathcal{M}}\frac{1}{|\mathbf{U}|}\sum_{X\in\mathbf{U}} q(m|X) log\ \frac{F^{'}(X_{=x}|m)}{F(X_{=x}|m)} \geqslant 0$$

This implies $L^{'} - L \geqslant 0$. □

We adapt the BN structure learning algorithm A1 so that it can learn from incomplete data. We use the EM-Mixture algorithm, i.e., Algorithm 6, to make the incomplete data complete in Phase 3. The whole algorithm will be referred to as A2. See Algorithm 7.

Algorithm 7 Modification of A1 algorithms for incomplete data (referred as A2)

```
1: Input:
2:     $\mathcal{D}$: incomplete dataset
3:     $m$: an integer representing the number of variables in $\mathcal{D}$
4: Output: Accepted sets of parents for each node
5: Phase 3: find the accepted $\mathbb{Q}_i$ for $X_i$
6:     $\mathbb{Q}_i$ is empty
7:     $S^* = \text{mscour}(X_i, \mathcal{D})$
8:     $\Pi_{X_i}$ is a parent set for $X_i$
9:     add $(X_i, \Pi_{X_i}, S^*)$ to $\mathbb{Q}_i$
10:    For each $X_k, k \in \{1, \ldots, m\}$ do:
11:        add $X_k$ to $\Pi_{X_i}$
12:        $S_{ki}$ = the BIC score of the updated $\Pi_{X_i}$
13:        if($S_{ki} > S^*$)
14:          add $(X_i, \Pi_{X_i}, S_{ki})$ to $\mathbb{Q}_k$
15:          For each $X_j, j \in \{1, \ldots, m\}, i \neq j \neq k$, do:
16:            add $X_j$ to $\Pi_{X_i}$
17:            if $\Pi_{X_i}$ is not complete:
18:              $\Pi'_{X_i}$ = EM.Mixture($\Pi_{X_i}$)
19:            else:
20:              $\Pi'_{X_i} = \Pi_{X_i}$
21:            $S_{ki}$ = the BIC score of the updated $\Pi'_{X_i}$
22:            if($S_{ki} > S^*$)
23:              add $(X_i, \Pi_{X_i}, S_{ki})$ to $\mathbb{Q}_k$
24:            else delete $X_j$ from $\Pi_{X_i}$
25:          end for
26:        else delete $X_k$ from $\Pi_{X_i}$
27:    end for
28:
```

3.3.2.2 Experiments

The experiments have been repeated ten times on ten different subsets for each MCAR rate on different models, using the generated datasets from the true models summarized in Table A.2 in A. We have compared our approach, denoted as A2, with three other methods. A1 denotes the BIC optimal learning from complete data created by omitting all rows containing a missing value. In [64], the authors proposed the soft and hard EM algorithms to fill in the missing values and learn an optimal BN structure from the completed data using Tabu search [65], which we refer to as A3 and A4, respectively.

The test scenarios, which include more than 700 incomplete datasets, are summarized in Figure 3.19. The resulting BNs of the simulations within each scenario are shown in Tables 3.14, 3.15, and 3.16.

The decision tree shown in Figure 3.19 is intended to guide practitioners regarding which imputation algorithm appears to perform the best, depending on the characteristics of their problem with incomplete data. Each leaf of the decision tree corresponds to a subset of the scenarios we studied, grouped according to the values of the experimental factors, to recommend which algorithm has the best average Structure Hamming Distance [66] (SHD) values between the essential graph of the learned model and the essential graph of the true model. The dominance of the algorithms has been tested using the Wilcoxon test [67]. We consider an algorithm to be better than another if it has a lower average SHD, and their confidence intervals do not overlap, i.e., the p-value of the Wilcoxon test is less than 5%.

In the results based on the SHD, A2 has achieved the best results. In the results based on the SHD and the Wilcoxon test, we have observed some important general trends:

- A2 appears to be a good algorithm in all scenarios.
- A2 is significantly better than other algorithms for Model M2 in Leaves B and G.
- A2 is significantly better than other algorithms for the model Child in Leaf C.
- A2 and A3 are significantly better than A1 and A4 for Models M1 in Leaves C, D, P, and K.
- A1 is significantly worse than other algorithms in all scenarios where the data size is smaller than 5,000.

Figure 3.20 represents the algorithm results for all models, dataset sizes, and MCAR rates.

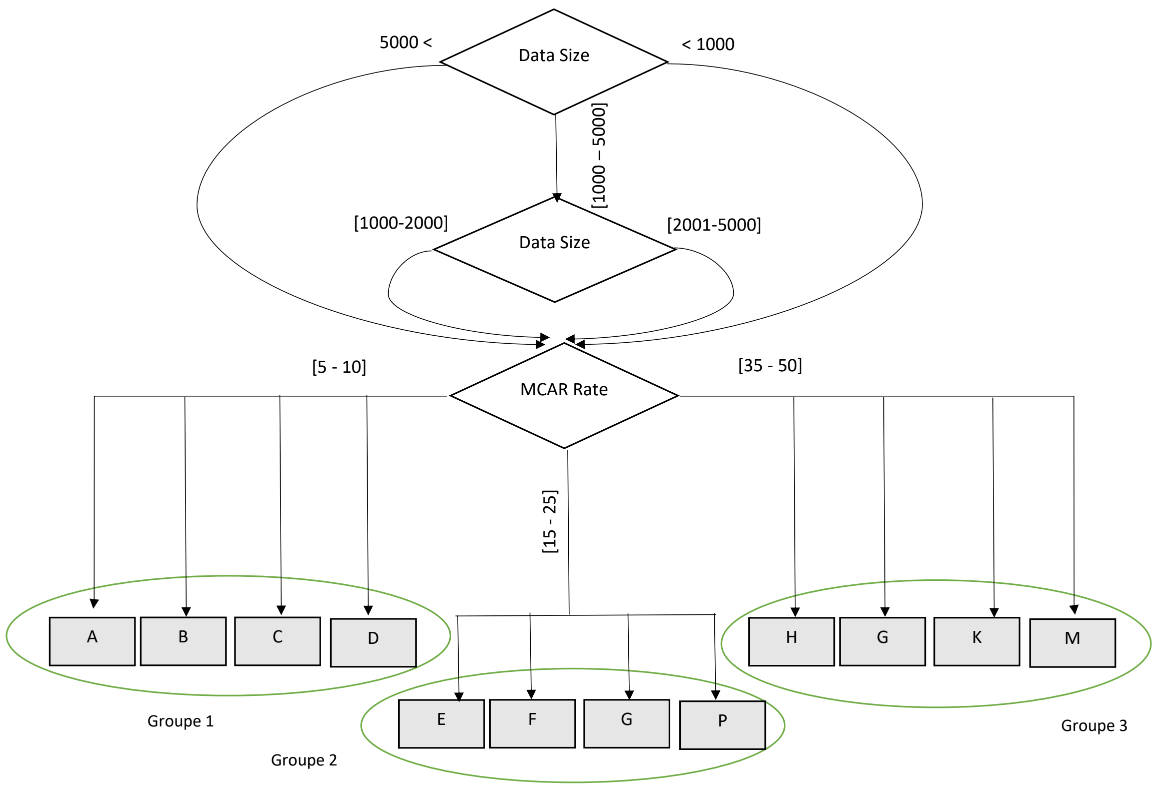


Figure 3.19: The decision tree for different test scenarios.

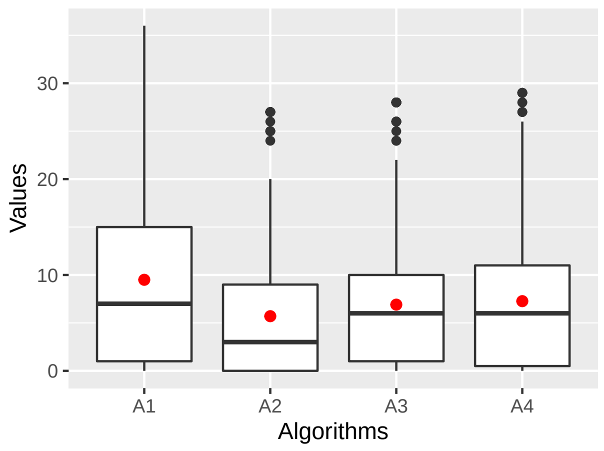


Figure 3.20: The Structural Hamming Distance to the true models from the resulting models of the structure learning algorithms using data generated from all models, summarized in Table A.2 averaged over all data sizes and all MCAR rates.

Table 3.14: Recommended algorithm by decision tree leaf where MCAR rate in [5 - 10] -Group 1.

Leaf	Size	Bayesian Network	Recommended Algorithm
A	Size >5000	Weather	A1, **A2**, A3, A4
		M1	**A2**, A3, A4
B	Size in [3000 - 5000]	Weather	**A2**, A3, A4
		M1	**A2**, A3, A4
		M2	**A2**
		Child	**A2**, A3, A4
C	Size in [1500 - 2500]	Weather	**A2**, A3, A4
		M1	**A2**, A3
		M2	**A2**, A3, A4
		Child	**A2**
D	Size <1000	Weather	**A2**, A3, A4
		M1	**A2**, A3
		M2	**A2**
		Child	**A2**, A3

Table 3.15: Recommended algorithm by decision tree leaf where MCAR rate in [15 - 25] - Group 2.

Leaf	Size	Bayesian Network	Recommended Algorithm
E	Size >5000	Weather	A1, **A2**, A3, A4
		M1	**A2**, A3, A4
F	Size in [3000 - 5000]	Weather	**A2**, A3, A4
		M1	**A2**, A3, A4
		M2	**A2**, A3
		Child	**A2**, A3
G	Size in [1500 - 2500]	Weather	**A2**, A3, A4
		M1	**A2**, A3, A4
		M2	**A2**
		Child	**A2**, A3
P	Size <1000	Weather	**A2**, A3, A4
		M1	**A2**, A3
		M2	**A2**
		Child	**A2**, A3

Table 3.16: Recommended algorithm by decision tree leaf where MCAR rate in [35 - 50] - Group 3.

Leaf	Size	Bayesian Network	Recommended Algorithm
H	Size >5000	Weather	A1, **A2**, A3, A4
		M1	**A2**, A3, A4
G	Size in [3000 - 5000]	Weather	**A2**, A3, A4
		M1	**A2**, A3
K	Size in [1500 - 2500]	Weather	**A2**, A3, A4
		M1	**A2**, A3
M	Size <1000	Weather	**A2**, A3, A4
		M1	**A2**

3.4 Belief Noisy-Or Model

The Belief Noisy-Or (BNO) model is suitable for describing a specific class of uncertain relationships in Bayesian networks [5] common in several practical applications of BNs. As an example, let us mention the QMR-DT network [68]. In Figure 3.21 we present the structure of a CPT $F(Y|X_1,\ldots,X_n)$ where auxiliary nodes $X'_1,\ldots,X'_n$ are added to explicitly separate the noisy relations from the logical OR relation. For a CPT with multiple parent variables $X_1,\ldots,X_n$ the noisy-or is defined as follows[5]:

$$\begin{aligned}
F(X_i^{'}=0|X_i=0) &= 1-\alpha\\
F(X_i^{'}=1|X_i=0) &= \alpha\\
F(X_i^{'}=0|X_i=1) &= p_i\\
F(X_i^{'}=1|X_i=1) &= 1-p_i
\end{aligned}$$

where $i\in\{1,\ldots,n\}$ and $p_i\in[0,1]$ is the parameter which defines the probability that the positive value x_i of variable X_i is inhibited – it is referred to as the inhibition probability and the parameter α specifies the possibility of a positive value even if the value of the corresponding parent variable is negative. In most experiments, we will set $\alpha=0$. The CPT of $F(Y|X'_1,\ldots,X'_n)$ represents the deterministic logical OR function, i.e.,

$$F(Y=0|X'_1=x'_1,\ldots,X'_n=x'_n) = \begin{cases}1 & \text{if } x'_1=0,\ldots,x'_n=0\\ 0 & \text{otherwise.}\end{cases}$$

Consequently, the CPT of $F(Y|X_1,\ldots,X_n)$, which represents the noisy-or function, is computed as follows:

$$\begin{aligned}
F(Y=0|X_1=x_1,\ldots,X_n=x_n) &= \prod_{i=1}^{n} F(X_i^{'}=0|X_i=x_i)\\
&= \prod_{1}^{n}(p_i)^{x_i}(1-\alpha)^{1-x_i}\\
F(Y=1|X_1=x_1,\ldots,X_n=x_n) &= 1-\prod_{1}^{n}(p_i)^{x_i}(1-\alpha)^{1-x_i}
\end{aligned}$$

where $i\in\{1,\ldots,n\}$ and $p_i\in[0,1]$ is the parameter which defines the probability that the positive value x_i of variable X_i is inhibited – it is referred to as the inhibition probability and the parameter α specifies the possibility of a positive value even if the value of the corresponding parent variable is negative. In most experiments, we will set $\alpha=0$. The CPT of $F(Y|X'_1,\ldots,X'_n)$ represents the deterministic logical OR function, i.e.,

$$F(Y=0|X'_1=x'_1,\ldots,X'_n=x'_n) = \begin{cases}1 & \text{if } x'_1=0,\ldots,x'_n=0\\ 0 & \text{otherwise.}\end{cases}$$

[5]In the case of one parent variable, we use probability values as specified in Table 3.17.

Consequently, the CPT of $F(Y|X_1, \dots, X_n)$, which represents the noisy-or function, is computed as follows:

$$F(Y = 0|X_1 = x_1, \dots, X_n = x_n) = \prod_{i=1}^{n} F(X_i^{'} = 0|X_i = x_i) = \prod_{1}^{n} (p_i)^{x_i} (1-\alpha)^{(1-x_i)} \tag{3.8}$$

$$F(Y = 1|X_1 = x_1, \dots, X_n = x_n) = 1 - F(Y = 0|X_1 = x_1, \dots, X_n = x_n) \tag{3.9}$$

Table 3.17: $F(X_i^{'}|X_i)$ table

		X_i	
		0	1
$X_i^{'}$	0	$1-\alpha$	0.2
	1	α	0.8

Table 3.18: N1 (Figure A.4): Marginal probability distributions

	C1	C2	C3	C4	C5	C6
$F(C_i = 0)$	.5	.6	.68	.744	.795	.837
$F(C_i = 1)$	.5	.4	.32	.256	.205	.163

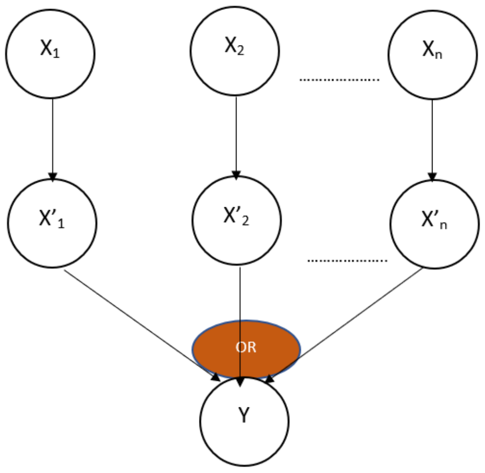


Figure 3.21: Noisy-or

3.4.1 Analysis of BNO Models

In this section we analyse the BNO models shown in Table A.3 in A where $\alpha = 0$. Tables 3.18, 3.19 and 3.20 show the marginal probability distributions (MPD) of the variables in the BNO models N1, N2 and BN2O respectively; see Figures A.4 and A.5. The tables illustrate the decrease in the marginal probability values for F$(C_i = 0)$ in the case of a node with more than one parent. See Table 3.20. This decrease is due to the properties of the product of probabilities in (3.8). On the other hand, they also illustrate the increase of this marginal probability with a higher number of its predecessors in previous layers; this increase depends on the number of layers above and also on the number of edges in these layers. See Table 3.18 and Table 3.19.

Table 3.19: N2 (Figure A.4): Marginal probability distributions

	C1	C2	C3	C4	C5	C6
$F(C_i = 0)$	.5	.6	.536	.539	.716	.707
$F(C_i = 1)$	.5	.4	.464	.461	.284	.293

Table 3.20: BN2O (Figure A.5): Marginal probability distributions

	C1	C2	C3	C4	C5	C6	C7	C8
$F(C_i = 0)$	.5	.5	.5	.5	.5	.129	.36	.36
$F(C_i = 1)$	.5	.5	.5	.5	.5	.871	.64	.64

Using the conditional probability distributions of the variables given their parents, we can easily calculate joint probability distributions $F(\mathbf{U})$ using formula (2.1) and conditional probability distributions (CPD) $F(\mathbf{X}_A|\mathbf{X}_B)$, where $\mathbf{X}_A \subseteq \mathbf{U}$ and $\mathbf{X}_B \subseteq \mathbf{U} \setminus \mathbf{X}_A$. Recall that a CPD for a particular configuration $\mathbf{x}_B$ of parent nodes $\mathbf{X}_B$ can be computed as[6]:

$$F(\mathbf{X}_A|\mathbf{X}_B = \mathbf{x}_B) = \frac{F(\mathbf{X}_A, \mathbf{X}_B = \mathbf{x}_B)}{F(\mathbf{X}_B = \mathbf{x}_B)} \tag{3.10}$$

The Kullback-Leibler Distance (KLD) of two conditional probability distributions $F(\mathbf{X}_A|\mathbf{X}_B)$ and $G(\mathbf{X}_A|\mathbf{X}_B)$ defined on the same state space is computed as the weighted average KLD of $F(\mathbf{X}_A|\mathbf{X}_B = \mathbf{x}_B)$ and $G(\mathbf{X}_A|\mathbf{X}_B = \mathbf{x}_B)$ over all configurations $\mathbf{x}_B$:

$$D(F(\mathbf{X}_A|\mathbf{X}_B)||G(\mathbf{X}_A|\mathbf{X}_B)) = \sum_{\mathbf{x}_B} F(\mathbf{X}_B = \mathbf{x}_B) \sum_{\mathbf{x}_A} F(\mathbf{X}_A = \mathbf{x}_A|\mathbf{X}_B = \mathbf{x}_B) \tag{3.11}$$

$$* \log \frac{F(\mathbf{X}_A = \mathbf{x}_A|\mathbf{X}_B = \mathbf{x}_B)}{G(\mathbf{X}_A = \mathbf{x}_A|\mathbf{X}_B = \mathbf{x}_B)} \tag{3.12}$$

$$= \sum_{\mathbf{x}_A, \mathbf{x}_B} F(\mathbf{X}_A = \mathbf{x}_A, \mathbf{X}_B = \mathbf{x}_B) \tag{3.13}$$

$$* \log \frac{F(\mathbf{X}_A = \mathbf{x}_A|\mathbf{X}_B = \mathbf{x}_B)}{G(\mathbf{X}_A = \mathbf{x}_A|\mathbf{X}_B = \mathbf{x}_B)} . \tag{3.14}$$

We will use KLD of conditional probability distributions estimated from the true data to support our arguments when we explain the results.

3.4.2 Experiments

We have performed experiments on different Belief noisy-or (BNO) models with their CPTs defined in Table 3.17 where $\alpha \in \{0, 0.2\}$ and the CPT of a node which has no

[6]Please, note that all BNs considered in this paper satisfy the condition $F(\mathbf{X}_B = \mathbf{x}_B) > 0$ for all $\mathbf{x}_B$.

parent is uniform, i.e., $F(X_i = 1|\{\}) = 0.5, F(X_i = 0|\{\}) = 0.5$. The experiments have been repeated ten times on ten different datasets generated from BNO models with different MCAR rates as specified in Table A.3 in Appendix A. In all Figures, we will denote additional edges by blue dashed lines, missing edges by red lines, and edges with different arrows by orange lines.

3.4.2.1 Model N1

The true N1 model is shown in Figure A.4 in Appendix A. We use this model as an example of a simple model with a chain structure. This model is motivated by some applications, e.g. from telecommunications. Let us summarise the results of this model:

- All algorithms learn the true structure when $\alpha \neq 0$ in all data sizes and all MCAR rates.
- Algorithms A2, A3 and A4 learn structures that differ from the true model in some cases when $\alpha = 0$, MCAR rate 15% and data size 1,000. For example, A3 and A4 learn an additional edge $C2 \rightarrow C4$, and A2 learns $C4 \rightarrow C6$ instead of $C5 \rightarrow C6$.
- Using the equation (3.10), we compute $F(C6|C5)$ and $F(C6|C4)$ from the true model N1. We found that their KLD value (calculated using equation (3.11)) is very small, it is only 0.001. Also, the Chi-square test of independence, whose p-value is less than 0.0001, shows that there is a strong dependence between $C6$ and $C4$, in addition to the relationship between $C6$ and $C5$ already explicitly present in the true model. Also, the BIC of the learned structure[7] is -2.252,93 and the BIC of the true model from the same dataset is -2.255,64. This can be explained by the deterministic conditional distribution $F(C6|C5 = 0)$ for $C5 = 0$. For these reasons we can conclude that we can accept that A2 learned $C4 \rightarrow C6$ instead of $C5 \rightarrow C6$.

3.4.2.2 Model N2

The true N2 model is shown on the right hand side of Figure A.4 in Appendix A. We use this model as an example of a model more complicated than the previous model N1. This model is motivated by some applications, e.g., by computer networks. We summarize the results of the experiments performed with this model:

- Figure 3.22 represents the Structure Hamming Distance (SHD) for all tested MCAR rates and models with $\alpha = 0$. We can observe that, as expected, the algorithm's performance is getting better with increasing the data size.
- We can see that A2 on average has a smaller SHD distance to the true model than other algorithms.

[7] We report the BIC value for one out of ten datasets as the results for the remaining nine are similar.

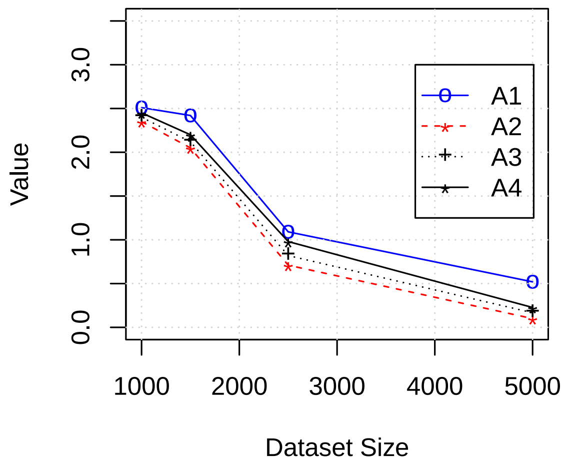
- In Figure 3.23, we compare the models learned from the datasets of size 5,000 with MCAR rate 10% using all four algorithms. We can see that A2 and A3 have the same SHD but they differ in that A3 has a missing edge $C4 \rightarrow C5$ while A2 has an additional edge $C3 \rightarrow C6$. This additional edge can be explained by observing that there is a chain of nodes $C3 \rightarrow C4 \rightarrow C6$ which the state 0 is propagated through because of $\alpha = 0$. In other words, we calculate $F(C3, C6|C1, C2, C4, C5)$ and the product $F(C3|C1, C2, C4, C5) \cdot F(C6|C4, C5)$ from the true model A.4 using equation (3.10). The KLD value (computed using equation (3.11)) of these two distributions is very small, it is only 0.02. Also, the chi-square test of independence of $C3$ and $C6$ reveals these variables are dependent (the test's p-value is smaller than 0.0001). The additional edge can be also supported by a comparison of BIC values of the learned structure with and without the additional edge $C3 \rightarrow C6$; they are -9,813.67 for the model with the additional edge and -9,880.5 for the true model.
- If $\alpha > 0$ then no additional edge is learned anymore, no matter what the MCAR rate is. Algorithms A2 and A4 we are always able to learn the true structure when the data size exceeds 1,000. Also, A1 and A3 learn the true structure when the data size is larger than 1,500.

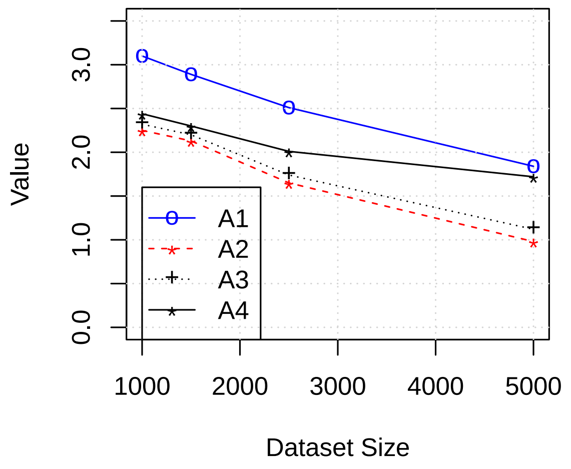
3.4.2.3 BN2O Models

These models are motivated by health care applications, such as the QMR-DT network [68]. We created 60 different BN2O models consisting of two layers with $N = 20$ nodes in total. They differ in the number of nodes in the first layer, namely $L_1 \in \{5, 8, 12, 15\}$; the number of nodes in the second layer is $L_2 = 20 - L_1$. The number of edges between layers is randomly generated with three different options $\frac{N}{2}$, $\frac{2 \cdot N}{2}$, and $\frac{4 \cdot N}{2}$; each option is repeated five times. Using these models, we generated several incomplete datasets with dataset sizes of 3,000 and 5,000 and MCARs of 10% and 15%. Figure 3.26 shows the boxplot of the number of additional and missing edges learned in all instances for each algorithm where the dataset size is 3,000 and for all MCAR rates. The results show that A2 performs better on average (i.e. the distance to the true model is smaller) than the other algorithms.

Next, a simpler example of a BN2O is discussed in more detail. The structure of this model is shown in Figure A.5 in Appendix A.

- Figure 3.24 shows the SHD of all learned models grouped by MCAR rates with models where $\alpha = 0$.
- The learned models from the data set of size 5,000, MCAR rate 10% and $\alpha = 0$ with all algorithms are shown in Figure 3.25. We can see that A2 performs better (i.e. the SHD distance to the true model is smaller) than the other algorithms.





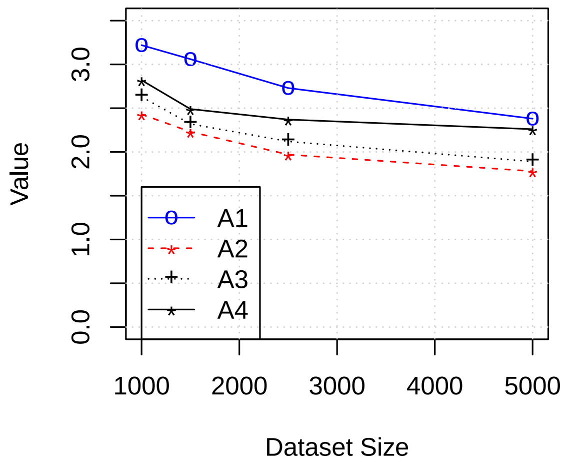


Figure 3.22: The Structural Hamming Distance of the resulting models of the structure learning algorithms to the true model (with $\alpha = 0$) using the data generated from the N2 model (the true model is presented in Figure A.4) using the average over ten experiments for different data sizes and for the MCAR rates of 5%, 10%, and 15%, respectively.

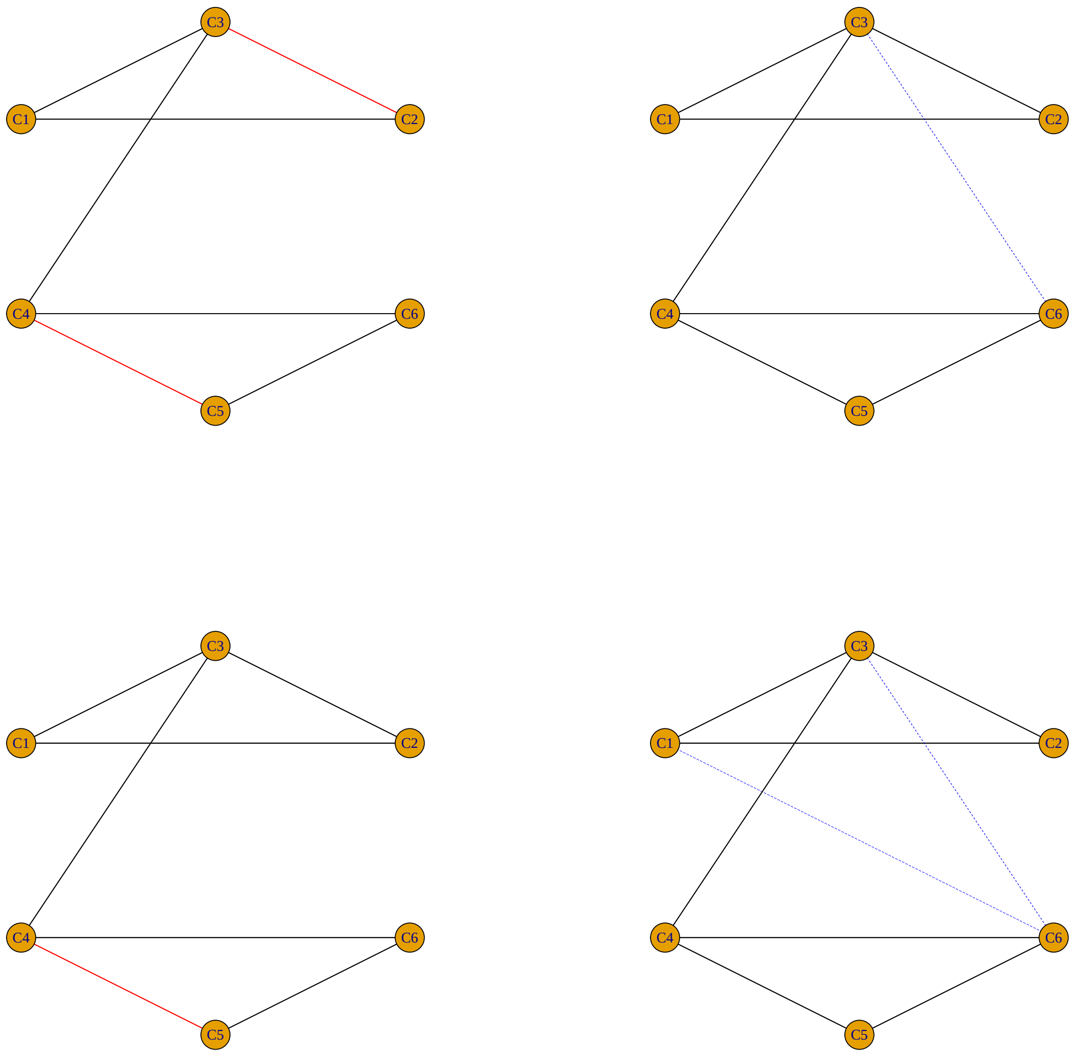


Figure 3.23: Models learned by A1, A2, A3, and A4, respectively, for most of ten datasets generated from the true N2 model (presented in Figure A.4) (for $\alpha = 0$) with the MCAR rate 10 and the data size of 5,000.

– Note the additional edge $C7 \rightarrow C8$ learned by A2 for most datasets. The argument for this extra edge is similar to that for the extra edge in the N2 model. Again we can see that the KLD of $F(C7, C8|C2, C5)$ and the product $F(C7|C2, C5).F(C8|C2, C5)$ is very small; it is only 0.002. Also, the chi-squared test of independence of $C7$ and $C8$ has a p-value smaller than 0.0001, and there is always a very small difference between the BIC of the model with the extra edge and the true model; for example, the BIC of the model with the extra edge is -7,331.8, while the BIC of the true model is -7,338.5 for one of the data sets generated.

– In the experiments with models with $\alpha > 0$, no extra edge was learned and the true model is successfully learned when the data size is 2,500 or greater for all MCAR rates.

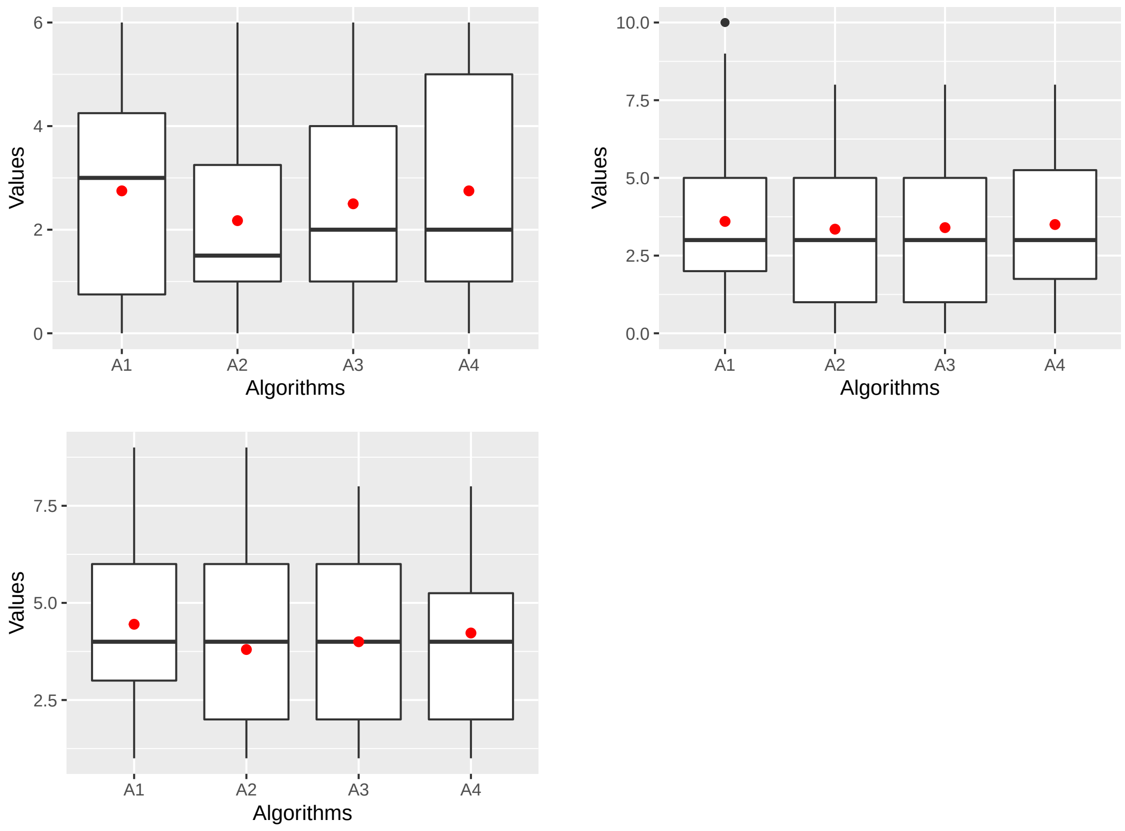


Figure 3.24: The Structural Hamming Distance to the true models of the resulting models of the structure learning algorithms using data generated from the BN2O model (the true model is presented in Figure A.5) (with $\alpha = 0$) averaged over all data sizes for MCAR rates of 5%, 10%, and 15%, respectively.

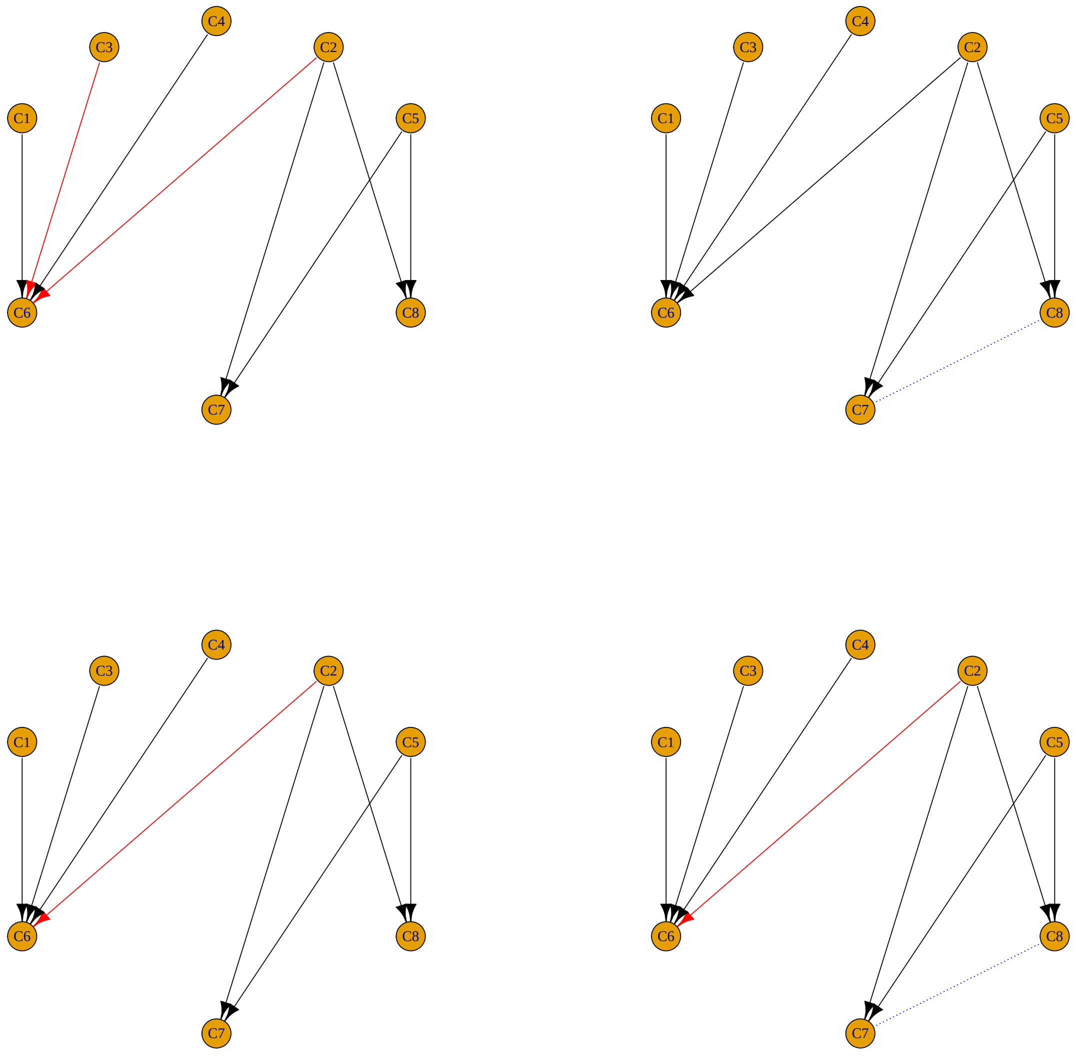


Figure 3.25: Models learned by A1, A2, A3, and A4, respectively, using data generated for most of ten datasets generated from the true BN2O model (presented in Figure A.5) (for $\alpha = 0$) with the MCAR rate 10 and the data size of 5,000.

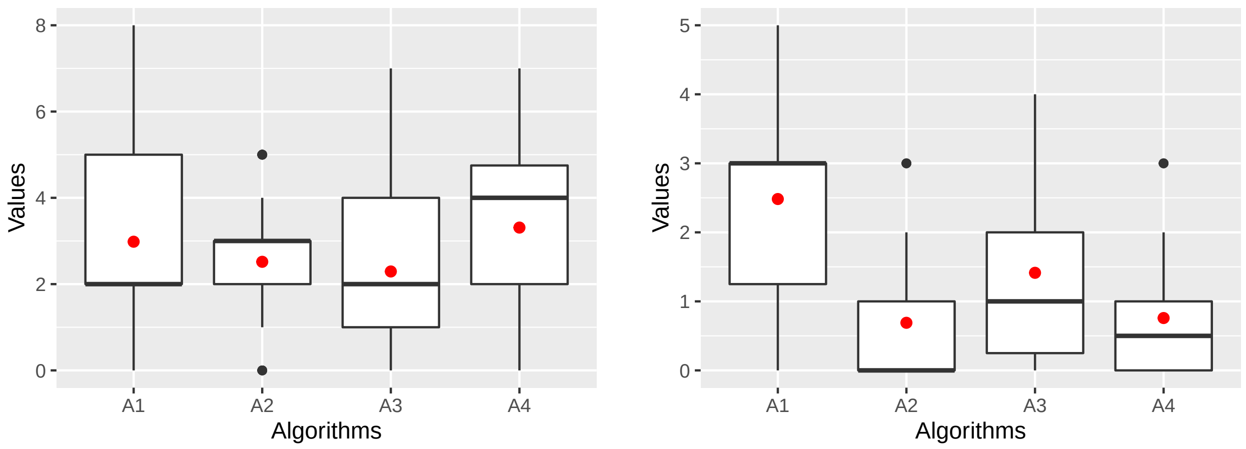


Figure 3.26: Results of the structure learning algorithms using data generated from the BN2O model (with $\alpha = 0$) with the data size of 3,000 and averaged over all tested MCAR rates. The plot on LHS displays the average number of additional edges and the plot on RHS displays the average number of missing edges.

3.4.2.4 Large BN2O Model

We have performed experiments with a model shown in Figure A.6 in Appendix A. This model consists of 25 variables; 14 in the first layer and 11 in the second layer. All algorithms required a data size of more than 5,000 to give a good performance. With the data size of 5,000 (and the MCAR rate of 10%) the achieved SHD of algorithms A1, A2, A3, and A4 still have not been very good – namely, 14.6, 11.2, 10, and 9.8, respectively. With the data size of 7,500 (and the MCAR rate of 10%) the achieved SHD of A1, A2, A3, and A4 are already much better – namely, 7.2, 4.6, 4.3, and 5.1, respectively. See Figure 3.27 for the learned models. With the data size of 12,000 we already get the true models except for the additional edges in the case of A2, as discussed in Section 3.4.2.3.

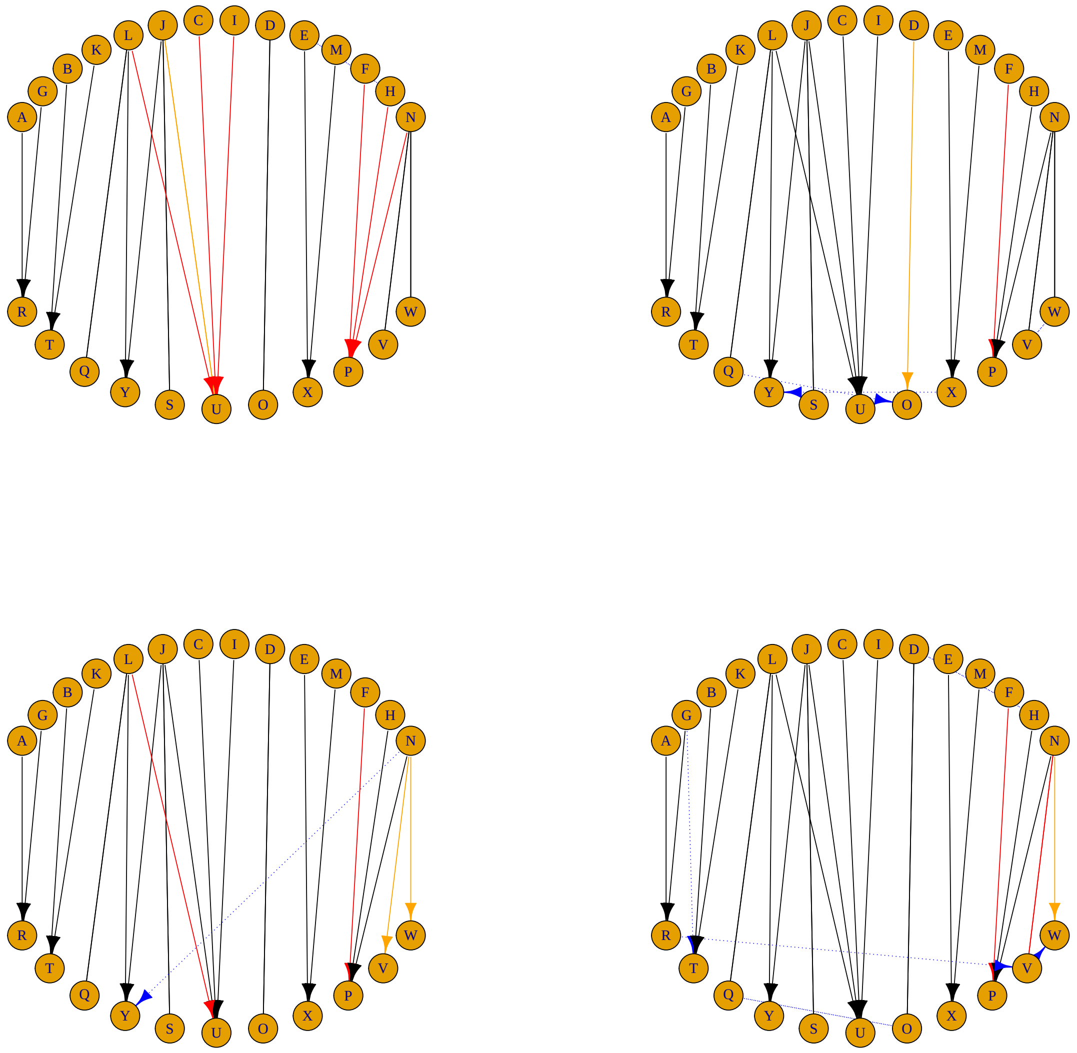


Figure 3.27: Models learned by A1, A2, A3, and A4, respectively, using the data generated from the large BN2O model consisting of 25 variables (for $\alpha = 0$) with the MCAR rate of 10% and the data size of 7,500 (true model is presented in Figure A.6).

3.5 Evaluation of a Novel Bayesian Network Model for Heart Disease Classification

This part of the thesis has been submitted for publication in [R6]. As shown in sections 3.3 and 3.4, the A2 Algorithm achieved the best experimental performance. Consequently, we will use this algorithm to analyze real public data from [69]. This dataset provides information about the diagnosis of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each patient is categorized as either normal or abnormal. The database consisted of 267 SPECT image sets (patients), from which we extracted features summarizing the original SPECT images. This extraction process resulted in 44 continuous feature patterns for each patient, which were further transformed into 22 binary feature patterns.

The CLIP3 algorithm [70] was used to generate classification rules from these patterns. The CLIP3 algorithm generated rules with an accuracy of 84.0% (compared to cardiologists' diagnoses) [69]. We refer to these data as D1.

In addition, we will use the real data explained in section 3.1. We refer to these data in this section as D2. Figure 3.28 shows the result of the A2 Algorithm compared to other algorithms.

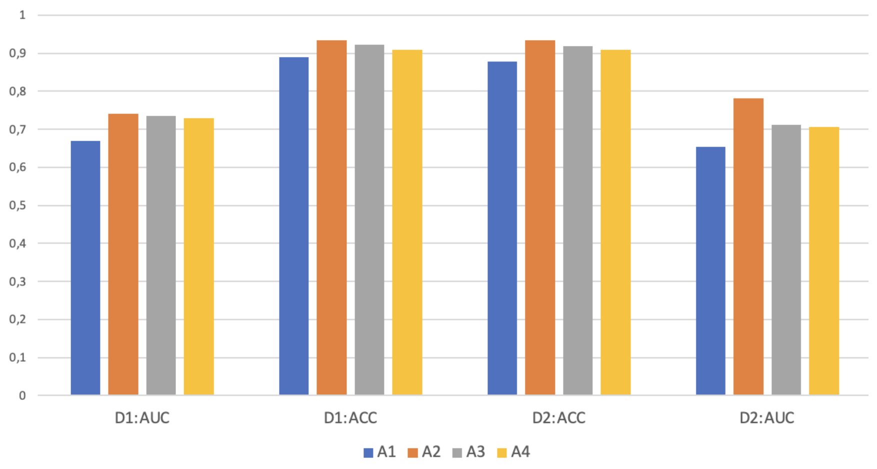


Figure 3.28: BNs - Results of experiments.

Figure 3.29 displays the Bayesian Network (BN) structure learned by A2. From this figure, we can discern that Cystatin C has a direct impact on Mortality. Cardiologists also employ this test as a marker for other health issues, such as kidney failure. To confirm Heart Attack Diagnosis, other markers come into play:

- Reduced Glomerular filtration rates (based on Cystatin C (GFCR) and MDRD (GRMD)) are associated with an increased mortality risk in patients with a heart attack. It is recommended for individuals with chronic heart disease.

- The C-reactive protein test predicts the likelihood of having heart disease. Elevated levels of C-reactive protein are associated with a three-fold greater risk of a heart attack.
- The circulating Cystatin C in blood is linked to an increased mortality risk in patients with heart disease. Typically, patients with higher circulating Cystatin C concentrations tend to be older and have a higher prevalence of systemic hypertension, resulting in a higher mortality risk.
- Creatinine (KREA) serves as a marker for diabetes and coronary artery disease (CAD) as well as kidney function. It is associated with an increased mortality risk in patients with a heart attack.

The learned structure illustrates the dependency between these markers and mortality. We also compared the classification performance of this model with the models evaluated in Section 3.1.3 using the D.Bin dataset. It achieved the highest accuracy of 93.146%. Moreover, its area under the ROC curve and F-Measure rank among the best achieved, with values of 0.718 and 0.252, respectively.

Furthermore, we report the Markov blanket (see Section 2.5.5) of the Mortality variable in the learned BN structure. The highlighted nodes in Figure 3.29 represent variables of this Markov blanket. The classification performance of the Markov blanket for this model is as follows: the accuracy is 92.6302%, the area under the ROC curve is 0.679, and the F-Measure is 0.1879.

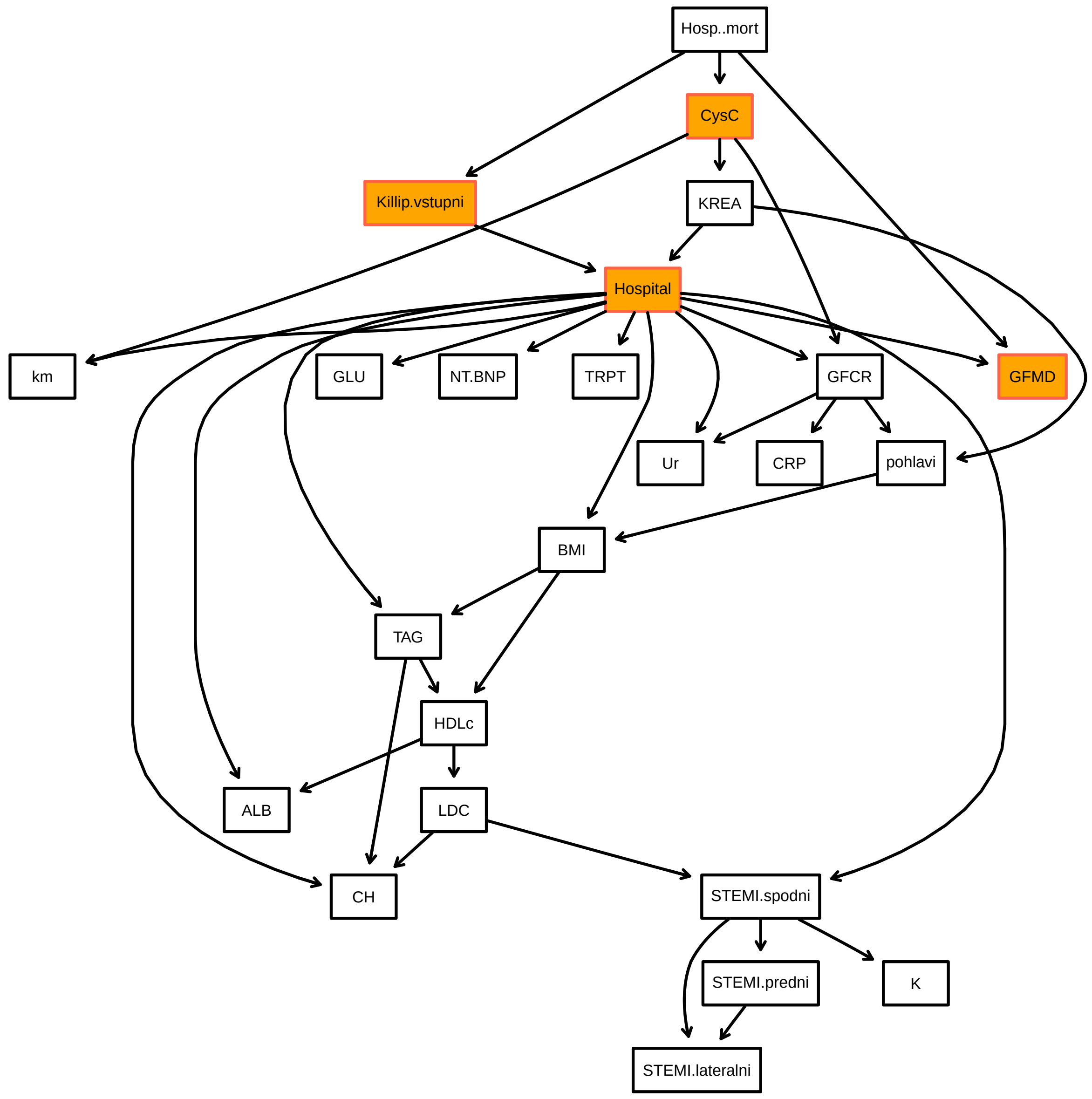


Figure 3.29: Bayesian Network learned from D.Bin using A2 specified in Algorithm 7.

3.5.1 BN Structure Analysis

We analyzed the learned BN structure to verify the relationship between AMI following cardiac normality/abnormality, its mortality, and important decision markers, which can help healthcare providers not only predict heart disease but also measure healthcare quality and reduce costs. In addition, the variables that directly affect the prediction results could be reviewed to identify factors that influence heart disease.

We believe that not all variables are related to the results of our study. We applied feature selection using Bayesian uncertainty [71, 72]. This is a common technique used in machine learning and statistics to select relevant features and eliminate irrelevant ones. The process typically involves estimating the uncertainty or importance of each feature using a Bayesian approach. There are various Bayesian-based feature selection methods, such as the Bayesian Information Criterion (BIC) [26] and Bayesian Regression [73]. We used the BIC approach with a threshold of 0.5.

We have identified the five most important variables. We provide valuable insights into the relationships and their impact on patient outcomes by analyzing the impact of each variable on the predictive power of the models. The variables are ranked from most important to least important according to the results of the BNs models: Killip, Cystatin (CYS), Glomerular filtration rate (GFMD), Kreatinin (KREA), and Albumin (ALB). The values are shown in Figure 3.30 where there predictive power of the models decreased between 6% with ALB up to 16% with Killip. The Table 3.21 represent in detail the impact of different variables on the predictive power of BNs.

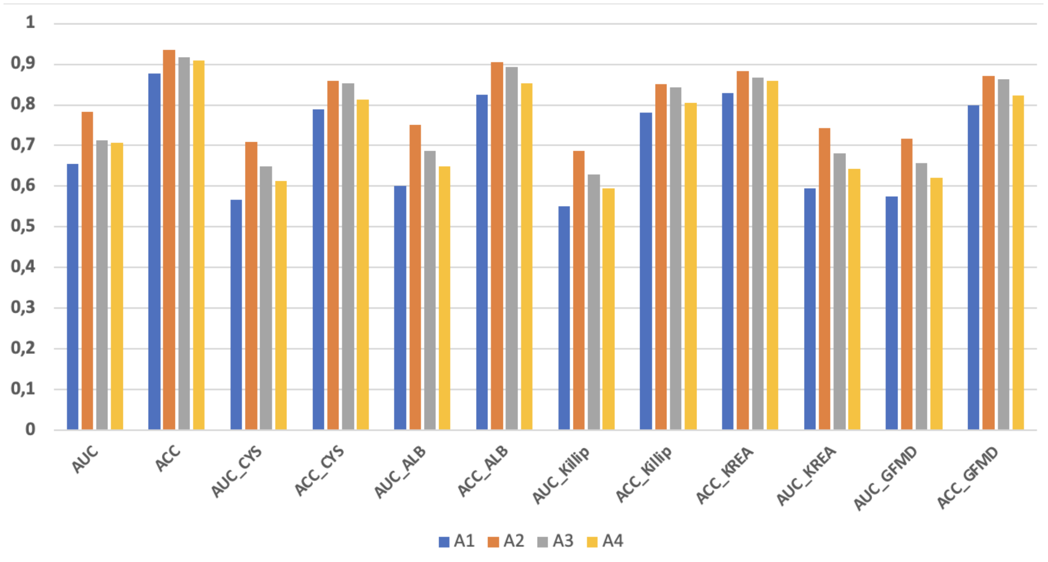


Figure 3.30: Impact of removing a variable on prediction qualitys

	A1	**A2**	**A3**	**A4**
AUC_CYS	13.27%	9.35%	8.90%	13.19%
ACC_CYS	10.14%	8.02%	7.08%	10.51%
AUC_ALB	8.06%	3.91%	3.44%	7.98%
ACC_ALB	5.92%	3.22%	2.62%	6.10%
AUC_Killip	15.87%	12.07%	11.63%	15.79%
ACC_Killip	11.09%	9.00%	8.07%	11.45%
ACC_KREA	5.50%	5.42%	5.53%	5.71%
AUC_KREA	9.08%	4.97%	4.50%	8.99%
AUC_GFMD	12.23%	8.26%	7.81%	12.15%
ACC_GFMD	9.06%	6.92%	5.97%	9.43%

Table 3.21: Impact of different nodes on the predictive power of BNs

Chapter 4

Conclusions

4.1 Summary

We used medical data from patients with AIM to compare the results of (a) basic statistical methods, (b) classification models, and (c) Bayesian networks modeling the relationships found in the data.

While some conclusions are specific to the data at hand, we also present general observations.

Certain attribute tests appear to be dependent on the class variable, although they may be conditionally independent. The observed dependence in the data may be attributed to their relationship with another attribute that, in turn, depends on the class variable. Detecting this using basic statistical methods can be challenging. In principle, Bayesian network (BN) learning algorithms can discover mediated correlations because they test not only pairwise independence but also conditional independence given the values of other variables.

Bayesian network structure learning often requires complete data. In this study, we introduced an adaptation of the learning process for the Chow-Liu, Tree Augmented Naive Bayes Classifier (TAN), and Selective TAN using incomplete and unbalanced medical datasets. These methods were successfully tested on our dataset. We found that our proposed algorithm (Algorithm 4) consistently outperformed the other algorithms.

In the reported experiments, we observed that the classifier based on the true model (TAN) consistently yielded the best results. our proposed algorithm (Algorithm 4) achieved the highest AUC with an MCAR rate higher than 20% for all data sizes greater than 1000. However, according to the Wilcoxon test, it's not significantly better than FL with an MCAR rate of 50%. FL achieved the highest AUC with an MCAR rate of less than 30%, but it's not better than Algorithm 4 according to the Wilcoxon test. In addition, STAN with a data size greater than 2000 and an MCAR rate of 40% recorded a better AUC than FL and bnclassify, but it is significantly better than them only with a dataset size of 10000 (significance level $\alpha = 0.05$).

Using the Wilcoxon signed-rank test on all mean AUC values, we can conclude that the proposed Algorithm 4 is significantly better than the other algorithms. FL is significantly better than STAN and TB. Comparing the structures of TAN shows that there is a significant difference between models 3.15 and 3.16 and the original, which means that the results concern the graph edges. Additionally, Algorithm 4 is significantly better than the other algorithms.

We proposed an approach to learn the optimal BN structure from incomplete data based on [10]. This adaptation imputes missing values using product mixtures learned by the EM algorithm [11].

We have shown that the sequence of log-likelihood values generated by the E-step and M-step of the EM algorithm is non-decreasing and that the algorithm converges.

Lemma 3.3.1 helps us to reduce the collection of candidate parent sets for a variable, which can speed up the learning algorithm.

We performed experiments on incomplete data generated from different types of BN models to compare the proposed algorithm 7 (A2) with other algorithms [10], soft and hard EM [64]. In our comparisons, we use the Structure Hamming Distance of the CPDAGs of the learned DAGs to the CPDAGs of the original models. These comparisons were performed on (a) general Bayesian networks and (b) Belief Noisy-or [5] (BNO) models with partially deterministic and non-deterministic conditional probability distributions. Experiments with type (b) models are motivated by the relationships in Bayesian networks that are common in practical applications of BNs.

We obtained the following results in the simulation studies.

General BN models:

(a) * A2 appears to be the best choice among the tested algorithms for learning the structure of BNs from any incomplete data whatever the data size and the missing MCAR rates are.

* In most scenarios corresponding to different data sizes and MCAR rates, A2 is significantly better than the other algorithms, and in no scenario. It is significantly worse than any other algorithm according to the Wilcoxon test.

(b) BNO models:

A2 is able to recover all true edges in the tested models except for the single chain model (shown in Figure A.4) at size 1,000 and the missing rate of 15%. However, the different learned structure of single chain model is justified by the Chi-square($\mathcal{X}^2$) test and the Kullback-Leibler distance (KLD) between the related conditional probabilities suggest there is a high degree of relationship between the connected variables. A2 has learned an additional edge in the case of Belief-Noisy-OR models (shown in Figure A.4) and BN2O (shown in Figure A.5). The additional edge is acceptable since the $\mathcal{X}^2$ test and KLD suggest there is a high degree of relationship between these variables. We have seen that BIC of the learned structure is almost equal to BIC of the true model. Similar behavior has been observed in other BNO models. A2 is always able to recover all edges with no missing one while other algorithms are not. The additional edges were justified by the Chi-square($\mathcal{X}^2$) test and the Kullback-Leibler distance (KLD) between the related conditional probabilities suggest there is a high degree of relationship between the connected variables. For large BN2O models, all algorithms require large data sizes to have a good performance; e.g., for the BN2O with 25 variables A2 needs at least 12,000 data records to learn the correct model (with the exception of additional edges).

We have empirically shown that our A2 behaves better than other tested algorithms on several studied BNs and in different scenarios. Based on these experiments, we can recommend this algorithm for practitioners that use BNs or BNOs with incomplete data.

Finally, the result of A2 using real medical data on patients with AIM is represent the dependency between important markers and mortality in exactly the same way as described by a cardiologist, which is done by analysing the impact of different nodes in the Bayesian Network on the prediction outcomes to identify factors affecting heart diseases. We have seen that the following variables Killip, Cystatin (CYS), Glomerular filtration rate (GFMD), Kreatinin (KREA) and Albumin (ALB) have the most impact on the predictive power.

4.2 Future Work

The author of the dissertation thesis suggests to explore the following:

- The results could be further improved by finding the optimal number of components to learn the EM-Mixture in Algorithm 6.
- The implementation of our methodology could be further improved to learn an optimal BN structure from large datasets by reducing the space of candidate parents.
- Our methodology could be further improved by learning the structure of large BN2O networks from incomplete data using constraint methods, and it could be applied to different real healthcare datasets.

Bibliography

[1] Murphy, K. P. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[2] Duda, R. O.; Hart, P. E.; Stork, D. G. Pattern classification second edition john wiley & sons. *New York*.

[3] Hastie, T.; Tibshirani, R.; Friedman, J. H.; et al. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[4] Koller, D.; Friedman, N. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[5] Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

[6] Murphy, K. P. A Brief Introduction to Graphical Models and Bayesian Networks 2. *http://www.cs.ubc.ca/˜murphyk/Bayes/bnintro.html*, 1998.

[7] Gruenerbl, A.; Osmani, V.; Bahle, G.; et al. Using smart phone mobility traces for the diagnosis of depressive and manic episodes in bipolar patients. In *Proceedings of the 5th Augmented human international conference*, 2014, pp. 1–8.

[8] Ertin, E.; Stohs, N.; Kumar, S.; et al. AutoSense: unobtrusively wearable sensor suite for inferring the onset, causality, and consequences of stress in the field. In *Proceedings of the 9th ACM conference on embedded networked sensor systems*, 2011, pp. 274–287.

[9] Clifton, D. A.; Niehaus, K.; Charlton, P.; et al. Health informatics via machine learning for the clinical management of patients. *Yearbook of medical informatics*, volume 24, no. 01, 2015: pp. 38–43.

[10] de Campos, C. P.; Zeng, Z.; Ji, Q. Structure learning of Bayesian networks using constraints. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 113–120.

[11] Grim, J.; Hora, J.; Boček, P.; et al. Statistical model of the 2001 Czech census for interactive presentation. *Journal of Official Statistics*, volume 26, no. 4, 2010: pp. 673–694.

[12] Friedman, N.; Geiger, D.; Goldszmidt, M. Bayesian Network Classifiers. *Machine Learning*, volume 20, no. 2-3, 1997: pp. 131–163.

[13] Lucas, P.; Boot, H.; Taal, B. Computer-based decision support in the management of primary gastric non-Hodgkin lymphoma. *Methods of information in medicine*, volume 37, no. 03, 1998: pp. 206–219.

[14] Shachter, R. D. Evaluating influence diagrams. *Operations research*, volume 34, no. 6, 1986: pp. 871–882.

[15] Friedman, N.; Linial, M.; Nachman, I.; et al. Using Bayesian networks to analyze expression data. *Journal of computational biology*, volume 7, no. 3-4, 2000: pp. 601–620.

[16] Ramoni, M.; Sebastiani, P.; Cohen, P. Bayesian clustering by dynamics. *Machine learning*, volume 47, no. 1, 2002: pp. 91–121.

[17] Margaritis, D. *Learning Bayesian Network Model Structure from Data. Ph.D. thesis.* USA: School of Computer Science, Carnegie-Mellon University. PA. Available as Technical Report CMU-CS-03-153, 2003.

[18] Pearl, J.; Verma, T. A theory of inferred causation. Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference. San Mateo, CA, USA: Morgan Kaufmann San Mateo, CA, 1991.

[19] Chickering, D. M. Learning Bayesian Networks is NP-Complete. In *Learning from Data*, Springer New York, 1996, pp. 121–130.

[20] Friedman, N.; Goldszmidt, M. *Learning Bayesian Networks with Local Structure.* Springer Netherlands, 1998, 421–459 pp.

[21] Cooper, G. An overview of the representation and discovery of causal relationships using Bayesian networks. *Computation, causation, and discovery*, 1999: pp. 4–62.

[22] Cooper, G.; Herskovits, E. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, volume 9, no. 4, 1992.

[23] Cussens, J. Bayesian network learning with cutting planes. arXiv preprint arXiv:1202.3713, 2012.

[24] Heckerman, D. *A Tutorial on Learning Bayesian Networks. Technical Report MSR-TR-95-06.* Redmond, WA: Microsoft Corporation, 1995.

[25] Lam, W.; Bacchus, F. Learning Bayesian Belief Networks: An approach based on the MDL Principle. *Computational Intelligence*, volume 10, no. 4, 1994: pp. 269–293.

[26] Liu, Z.; Malone, B.; Yuan, C. Empirical evaluation of scoring functions for Bayesian network model selection. In *Proceedings of the Ninth Annual MCBIOS Conference. Dealing with the Omics Data Deluge*, Oxford, MS, USA.: BMC Bioinformatics, 2012.

[27] Song, P. C.; Chong, H. Y.; Ong, H. C.; et al. A Model of Bayesian Network Analysis of The Factors Affecting Student's Higher Level Study Decision: The Private Institution Case. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, volume 8, no. 2, 2016: pp. 105–109.

[28] Verma, T. S.; Pearl, J. Computer Science Today. Recent Trends and Developments. In Uncertainty in Artificial Intelligence (UAI). Elsevier Science publishers B.V. (North-Holland). San Francisco, 1990, pp. 220–227.

[29] Weinberger, N. Faithfulness, Coordination and Causal Coincidences. *Erkenntnis*, volume 83, 2018.

[30] de Campos, C.; Mauro, S.; Giorgio, C.; et al. Entropy-based pruning for learning Bayesian networks using BIC. *Artificial Intelligence*, volume 260, 2018: pp. 42–50.

[31] Verma, T. S.; Pearl, J. *Equivalence and synthesis of causal models.* UCLA, Computer Science Department, 1991.

[32] Spirtes, P.; Glymour, C. N.; Scheines, R.; et al. *Causation, prediction and search.* MIT press, 2000.

[33] Scutari, M. Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimised Implementations in the bnlearn R Package. *CoRR*, volume abs/1406.7648, 2014.

[34] Meek, C. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Canada: Morgan Kaufmann, 1995, pp. 403–410.

[35] Chickering, D. M. A Transformational Characterization of Equivalent Bayesian Network Structures. In *In Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI95)*, CAUnited States: Morgan Kaufmann, 1995, pp. 87–98.

[36] Chen, X.; Guo, W.; Zhao, L.; et al. Acute myocardial infarction detection using deep learning-enabled electrocardiograms. *Frontiers in cardiovascular medicine*, volume 8, 2021.

[37] Mandair, D.; Tiwari, P.; Simon, S.; et al. Prediction of incident myocardial infarction using machine learning applied to harmonized electronic health record data. *BMC medical informatics and decision making*, volume 20, no. 1, 2020: pp. 1–10.

[38] Kaufmann, T.; Forte, J. C.; Hiemstra, B.; et al. A Bayesian Network analysis of the diagnostic process and its accuracy to determine how clinicians estimate cardiac function in critically ill patients: prospective observational cohort study. *JMIR medical informatics*, volume 7, no. 4, 2019: p. e15358.

[39] Krumholz, H. M.; Normand, S.-L. T.; Galusha, D. H.; et al. Risk-Adjustment Models for AMI and HF 30-Day Mortality, Methodology. Technical report, Harvard Medical School, Department of Health Care Policy, 2007.

[40] Vomlel, J.; Kružík, H.; Tůma, P.; et al. Machine Learning Methods for Mortality Prediction in Patients with ST Elevation Myocardial Infarction. In *In the Proceedings of The Nineth Workshop on Uncertainty Processing WUPES'12*, 2012, p. 204—213.

[41] Wasserman, L. *All of Statistics*. Springer-Verlag New York, 2004.

[42] Hall, M.; Frank, E.; Holmes, G.; et al. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, volume 11, no. 1, 2009: pp. 10–18.

[43] Quinlan, R.; Kaufmann, M. C4.5: Programs for Machine Learning. *Machine Learning*, volume 29, 1993: pp. 131–163.

[44] Le Cessie, S.; Van Houwelingen, J. C. Ridge estimators in logistic regression. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, volume 41, no. 1, 1992: pp. 191–201.

[45] Duda, R. O.; Hart, P. E. Pattern Classification and Scene Analysis. *Wiley-Interscience, Oxford*, volume 30, 1973: pp. 106–10.

[46] Cooper, G. F. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, volume 9, 1992: pp. 309 – 347.

[47] Greiner, R.; Zhou, W. Structural Extension to Logistic. *Regression: Discriminative Parameter Learning of Belief Net, Classifiers., AAAI*, 2002.

[48] Vrigkas, M.; Kazakos, E.; Nikou, C.; et al. Human Activity Recognition Using Robust Adaptive Privileged Probabilistic Learning. *arXiv preprint arXiv:1709.06447*, 2017.

[49] François, O.; Leray, P. Learning the Tree Augmented Naive Bayes Classifier from incomplete datasets. In *Probabilistic Graphical Models*, Citeseer, 2006, pp. 91–98.

[50] Mihaljevic, B.; Bielza, C.; Larranaga, P. *Comments on bnclassify package runtimes, URl: http://127.0.0.1:15009/library/bnclassify/doc/runtimes.pdf.* 2015.

[51] Højsgaard, S. Bayesian networks in R with the gRain package. *Journal of Statistical Software*, volume 46, no. 10, 2012: pp. 1–26.

[52] Chow, C.; Liu, C. Approximating discrete probability distributions with dependence trees. *IEEE Trans, on Info, Theory*, volume 14, 1968: p. 462–467.

[53] Blanco, R.; Inza, I.; Merino, M.; et al. Feature selection in Bayesian classifiers for the prognosis of survival of cirrhotic patients treated with TIPS. *Journal of Biomedical Informatics*, volume 38, no. 5, 2005: pp. 376–388.

[54] Laza, R.; Pavón, R.; Reboiro-Jato, M.; et al. Evaluating the effect of unbalanced data in biomedical document classification. *Journal of integrative bioinformatics*, volume 8, no. 3, 2011: pp. 105–117.

[55] Rahman, M. M.; Davis, D. N. Addressing the class imbalance problem in medical datasets. *International Journal of Machine Learning and Computing*, volume 3, no. 2, 2013: p. 224.

[56] Alameddine, I.; Cha, Y.; Reckhow, K. H. An evaluation of automated structure learning with Bayesian networks: an application to estuarine chlorophyll dynamics. *Environmental Modelling & Software*, volume 26, no. 2, 2011: pp. 163–172.

[57] Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; et al. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, volume 16, 2002: pp. 321–357.

[58] Jamshidian, M.; Jalal, S.; Jansen, C. MissMech: An R Package for Testing Homoscedasticity, Multivariate Normality, and Missing Completely at Random (MCAR). *Journal of Statistical Software*, volume 56, no. 6, 2014.

[59] de Campos, C. P.; Qiang, J. Efficient structure learning of Bayesian networks using constraints. *The journal of Machine Learning Research*, volume 12, 2011: pp. 663–689.

[60] Dempster, A. P.; Laird, N. M.; Rubin, D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Roy. Statist. Soc. B*, volume 39, 1977: pp. 1–38.

[61] Grim, J.; Bocek, P. Statistical model of Prague households for interactive presentation of census data. In *SoftStat 95. Advances in Statistical Software 5. Conference on the Scientific Use of Statistical Software*, Heidelberg, DE, 1996.

[62] Scrucca, L.; Fop, M.; Murphy, T. B.; et al. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, volume 8, no. 1, 201: pp. 289–31.

[63] Vajda, I. *Theory of Statistical Inference and Information.* Netherlands: Springer, 1989.

[64] Ruggieri, A.; Stranieri, F.; Stella, F.; et al. Hard and Soft EM in Bayesian Network Learning from Incomplete Datas. *Algorithms*, volume 13, no. 12, 2020: pp. 329–356.

[65] Glover, F. Tabu search-part I. *ORSA Journal on computing*, volume 1, no. 3, 1989: pp. 190–206.

[66] Scutari, M.; Denis, J. *Bayesian Networks: with Examples in R*. Boca Raton: Chapman & Hall, 2014.

[67] Neuhauser, M. International Encyclopedia of Statistical Science. Canada: Springer Berlin Heidelberg, 2011, pp. 1656–1658.

[68] Shwe, M. A.; Middleton, B.; Heckerman, D. E.; et al. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. *Methods of Information in Medicine*, volume 30, no. 04, 1991: pp. 241–255.

[69] Cios, K. L., Krzysztof; Goodenday, L. SPECT Heart. UCI Machine Learning Repository, 2001, DOI: https://doi.org/10.24432/C5P304.

[70] Cios, K. J.; Wedding, D. K.; Liu, N. CLIP3: Cover learning using integer programming. *Kybernetes*, volume 26, 1997: pp. 513–536. Available from: https://api.semanticscholar.org/CorpusID:62721886

[71] Inza, I.; Larrañaga, P.; Etxeberria, R.; et al. Feature subset selection by Bayesian network-based optimization. *Artificial intelligence*, volume 123, no. 1-2, 2000: pp. 157–184.

[72] Goldstein, O.; Kachuee, M.; Karkkainen, K.; et al. Target-focused feature selection using uncertainty measurements in healthcare data. *ACM Transactions on Computing for Healthcare*, volume 1, no. 3, 2020: pp. 1–17.

[73] West, M. Outlier models and prior distributions in Bayesian linear regression. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, volume 46, no. 3, 1984: pp. 431–439.

[74] Abramson, B.; Brown, J.; E, W.; et al. Hailfinder: A Bayesian system for forecasting severe weather. *International Journal of Forecasting*, volume 12, no. 1, 1996: pp. 57–71, ISSN 0169-2070, doi:https://doi.org/10.1016/0169-2070(95)00664-8, probability Judgmental Forecasting. Available from: https://www.sciencedirect.com/science/article/pii/0169207095006648

[75] Dawid, A. P. Prequential analysis, stochastic complexity and Bayesian inference. *Bayesian statistics*, volume 4, 1992: pp. 109–125.

Author's Publications Relevant to the Thesis:

[R1] Salman, I. and Vomlel, J. *A machine learning method for incomplete and imbalanced medical data.* In: 20TH Czech-Japan seminar on data analysis and decision making 2017

The paper has been cited in:

- Web of Science:
  - Saturi, S. *Review on Machine Learning Techniques for Medical Data Classification and Disease Diagnosis. Regen. Eng. Transl. Med. (2022). https://doi.org/10.1007/s40883-022-00273-y*

[R2] Alnader, A.L.I., Salman, I., Ajami, K., and Alzein, A. *Arabic ontology-based approach for chest diseases diagnosis.* Journal of Theoretical and Applied Information Technologythis link is disabled, 2018, 96(21), pp. 7077–7087

[R3] Salman, I. *Heart attack mortality prediction: An application of machine learning methods.* Turkish Journal of Electrical Engineering and Computer Sciences, 2019, 27(6), pp. 4378–4389

The paper has been cited in:

- Web of Science:
  - Ahsan, M. M., and Siddique, Z. (2022). *Machine learning-based heart disease diagnosis: A systematic literature review.* Artificial Intelligence in Medicine, 102289.
  - Bhardwaj, P., Bhandari, G., Kumar, Y., and Gupta, S. (2022). *An Investigational Approach for the Prediction of Gastric Cancer Using Artificial Intelligence Techniques: A Systematic Review.* Archives of Computational Methods in Engineering, 1-22.

- ∗ Alluri, R. (2021). *Machine learning for health care.* Blockchain and Machine Learning for e-Healthcare Systems, 319-342.
- ∗ Stiglic, G., Kocbek, P., Fijacko, N., Zitnik, M., Verbert, K., Cilar, L. (2020). *Interpretability of machine learning-based prediction models in healthcare.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(5), e1379.

– Scopus:

- ∗ Bah, I. (2022). *KNN Algorithm Used for Heart Attack Detection.* FES Journal of Engineering Sciences, 11(1), 7-19.
- ∗ Tadiparthi, P. K., and Kuna, V. (2022). *Heart Disease Prediction Using Machine Learning Algorithms: A Systematic Survey.*
- ∗ Keya, M. S., Shamsojjaman, M., Hossain, F., Akter, F., Islam, F., and Emon, M. U. (2021, March). *Measuring the Heart Attack Possibility using Different Types of Machine Learning Algorithms.* In 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS) (pp. 74-78). IEEE.
- ∗ Yan, Z., and Zong, L. (2020, July). *Spatial prediction of housing prices in Beijing using machine learning algorithms.* In Proceedings of the 2020 4th high performance computing and cluster technologies conference. 2020 3rd international conference on big data and artificial intelligence (pp. 64-71).

[R4] Salman, I. *Learning the Structure of the Tree and Tree-Augmented Naive Bayesian from Incomplete and Impalanced data.* In: International Arab Conference on Information Technology. IEEE Xplore, 2020. p. 1-7. ISSN 1812-0857. ISBN 978-1-7281-8855-3.

[R5] Salman, I. and Vomlel, J. *Learning The Structure of Bayesian Network from Incomplete Data Using a Mixture Model.* Informatica, 47(1).

The paper has been cited in:

– Web of Science:

- ∗ Kwiendacz, H. Wijata, A. M., Nalepa, J., Piaśnik, J. et al. *Machine learning profiles of cardiovascular risk in patients with diabetes mellitus: the Silesia Diabetes-Heart Project. Cardiovascular Diabetology, 22(1), 218.*

[R6] Salman, I. Ganapati P. and Vomlel, J. *Development and Performance Evaluation of a Novel Bayesian Network Model for the Classification of Heart Disease.* Applied Clinical Informatics (Resubmitted: Under review)
.

Other Publications of the Author

[O1] Daood, A., Salman, I., and Ghneim, N. *Comparison study of automatic classifiers performance in emotion recognition of Arabic social media users.* Journal of Theoretical and Applied Information Technology, 2017, 95(19), pp. 5172–5183

The paper has been cited in:

- Scopus:
  - Baali, M., and Ghneim, N. (2019). *Emotion analysis of Arabic tweets using deep learning approach.* Journal of Big Data, 6(1), 1-12.
  - Cassab, S., and Kurdy, M. B. (2020). *Ontology-based emotion detection in arabic social media.* International Journal of Engineering Research Technology (IJERT), 9(08), 1991-2013.

Appendix A

Appendix

Simulation Scenarios

This Appendix provides an inclusive list of all experiments in the simulation study described in Sections 3.2.4, 3.3.2.2 and 3.4.2, organized by their main characteristics in Tables A.1, A.2 and A.3, respectively. The number of components in each experiment in the Sections 3.3.2.2 and 3.4.2 selected based on the number of variables in the datasets. The true models mentioned in the Table A.2 are shown in Figure A.3. The true models mentioned in the Table A.3 are shown in Figures A.4 and A.5.

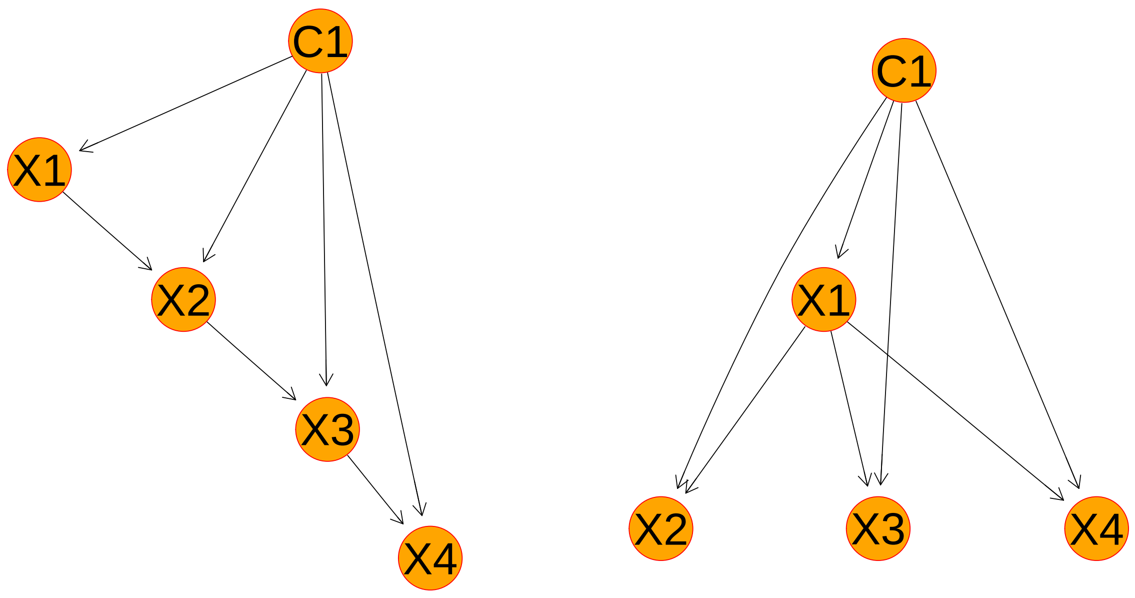


Figure A.1: T1 and T2 true models, respectively.

Table A.1: Description of the key factors of all TAN experiments in the simulation study.

Network	Missing Rate (MCAR)	Replicates	Sample Size
T1(Figure A.1)	10	10	1000,2000,5000,7000,10000
	20	10	1000,2000,5000,7000,10000
	30	10	1000,2000,5000,7000,10000
	40	10	1000,2000,5000,7000,10000
	50	10	1000,2000,5000,7000,10000
T2 (Figure A.1)	10	10	1000,2000,5000,7000,10000
	20	10	1000,2000,5000,7000,10000
	30	10	1000,2000,5000,7000,10000
	40	10	1000,2000,5000,7000,10000
	50	10	1000,2000,5000,7000,10000
T3 (Figure A.2)	10	10	1000,2000,5000,7000,10000
	20	10	1000,2000,5000,7000,10000
	30	10	1000,2000,5000,7000,10000
	40	10	1000,2000,5000,7000,10000
	50	10	1000,2000,5000,7000,10000
T4 (Figure A.2)	10	10	1000,2000,5000,7000,10000
	20	10	1000,2000,5000,7000,10000
	30	10	1000,2000,5000,7000,10000
	40	10	1000,2000,5000,7000,10000
	50	10	1000,2000,5000,7000,10000

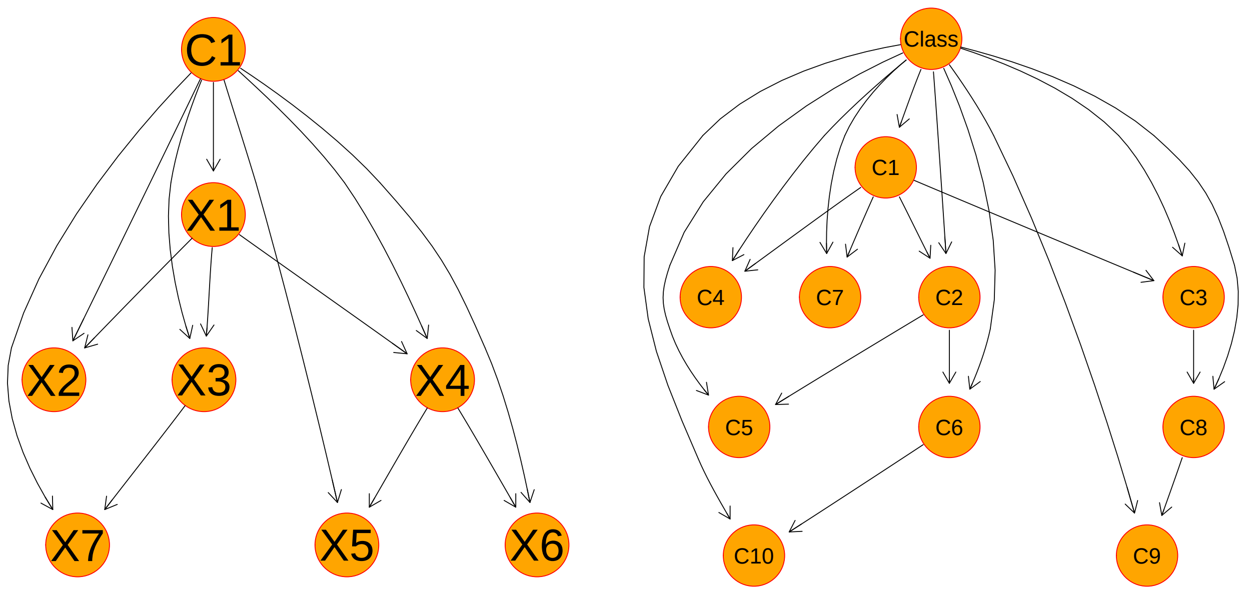


Figure A.2: T3 and T4 true models, respectively.

Table A.2: Description of the key factors of all BN experiments in the simulation study.

Network	Missing Rate (MCAR)	Replicates	Sample Size
Weather [74]	10	10	100, 500,1000,5000,10000
	25	10	100, 500,1000,5000,10000
	50	10	100, 500,1000,5000,10000,13000
Child [75]	10	10	1000, 2000,3000,5000
	15	10	1000, 2000,3000,5000
	50	10	1000, 2000,3000,5000
M2 (Figure A.3)	5	10	500,1000,1500,2500,5000
	10	10	500,1000,1500,2500,5000
	15	10	500,1000,1500,2500,5000
	25	10	500,1000,1500,2500,5000
M1 (Figure A.3)	10	10	500,1500,2500,5000,10000,13000
	20	10	500,1500,2500,5000,10000,13000
	35	10	500,1500,2500,5000,10000,13000
	50	10	500,1500,2500,5000,10000,13000

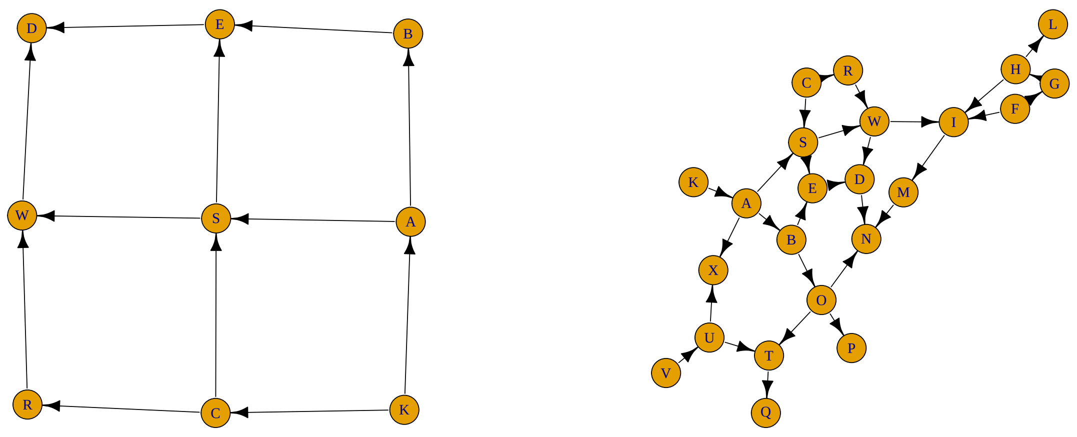


Figure A.3: M1 and M2 true models, respectively

Table A.3: Description of the key factors of all Belief Noisy-OR experiments in the simulation study (true models are presented in Figures A.4 and A.5).

Network	Missing Rate (MCAR)	Replicates	Sample Size
BN2O	5	10	1000,1500,2500,5000
	10	10	1000,1500,2500,5000
	15	10	1000,1500,2500,5000
N1	5	10	1000,1500,2500,5000
	10	10	1000,1500,2500,5000
	15	10	1000,1500,2500,5000
N2	5	10	1000,1500,2500,5000
	10	10	1000,1500,2500,5000
	15	10	1000,1500,2500,5000
large BN2O	10	10	5000, 7500

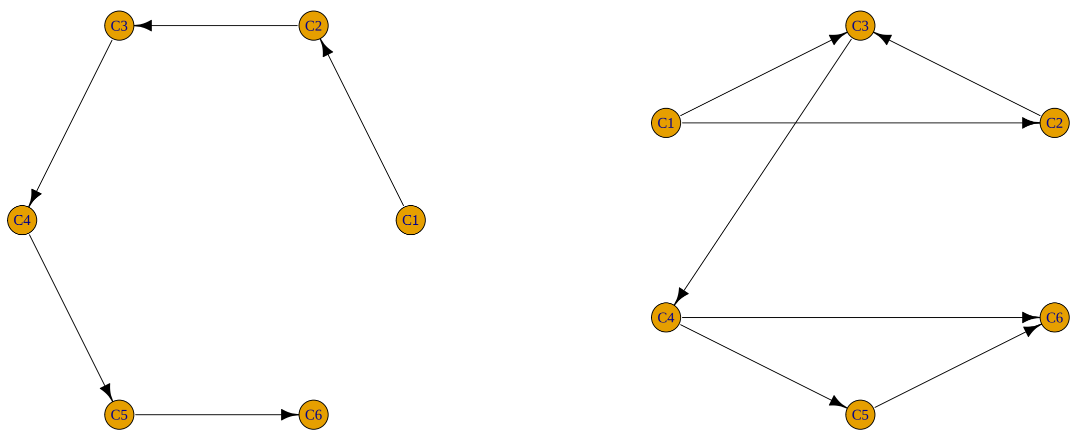


Figure A.4: N1 and N2 true models, respectively. Their marginal probability distributions are summarized in Tables 3.18 and 3.19

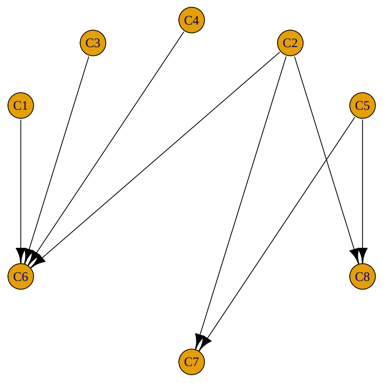


Figure A.5: BN2O true model. Its marginal probability distributions are summarized in Table 3.20

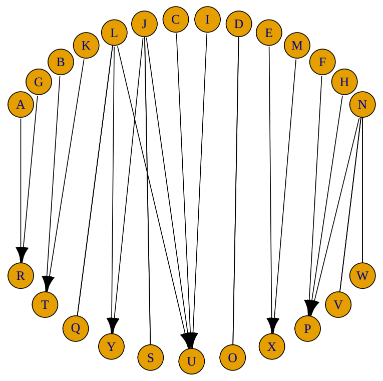


Figure A.6: Example of a large BN2O model with 25 variables (whose learned models are presented in Figure 3.27).

Algorithm2

All algorithms are available at https://github.com/issamsalman/PhD-thesis-algorithms.

The computational complexity of A2 algorithm is as follows:

- The computational complexity of A2 depends on the number of components, data size, the convergence criteria, etc.
- Generally, the complexity of the A2 Algorithm is $O(T \cdot C)$, where T is the number of iterations and C represents the complexity of each iteration.
- C is number of data point multiplied by number of components.