The classification performance of Bayesian Networks Classifiers: A case study of detecting Denial of Service (DoS) attacks in cloud computing environments

By

Lindani Moyo

[BSc Honours]

A dissertation submitted in fulfillment of the requirements of the Degree of Master of Science

in

Computer Science

Department of Computer Science

Faculty of Science & Agriculture

University of Fort Hare

Together in Excellence

Supervised

by

Prof Khulumani Sibanda

[November 2015]

Abstract

In this research we propose a Bayesian networks approach as a promissory classification technique for detecting malicious traffic due to Denial of Service (DoS) attacks. Bayesian networks have been applied in numerous fields fraught with uncertainty and they have been proved to be successful. They have excelled tremendously in classification tasks i.e. text analysis, medical diagnoses and environmental modeling and management. The detection of DoS attacks has received tremendous attention in the field of network security. DoS attacks have proved to be detrimental and are the bane of cloud computing environments. Large business enterprises have been/or are still unwilling to outsource their businesses to the cloud due to the intrusive tendencies that the cloud platforms are prone too. To make use of Bayesian networks it is imperative to understand the "ecosystem" of factors that are external to modeling the Bayesian algorithm itself. Understanding these factors have proven to result in comparable improvement in classification performance beyond the augmentation of the existing algorithms. Literature provides discussions pertaining to the factors that impact the classification capability, however it was noticed that the effects of the factors are not universal, they tend to be unique for each domain problem. This study investigates the effects of modeling parameters on the classification performance of Bayesian network classifiers in detecting DoS attacks in cloud platforms. We analyzed how structural complexity, training sample size, the choice of discretization method and lastly the score function both individually and collectively impact the performance of classifying between normal and DoS attacks on the cloud. To study the aforementioned factors, we conducted a series of experiments in detecting live DoS attacks launched against a deployed cloud and thereafter examined the classification performance in terms of accuracy of different classes of Bayesian networks. NSL-KDD dataset was used as our training set. We used ownCloud software to deploy our cloud platform. To launch DoS attacks, we used hping3 hacker friendly utility. A live packet capture was used as our test set. WEKA version 3.7.12 was used for our experiments. Our results show that the progression in model complexity improves the classification performance. This is attributed to the increase in the number of attribute correlations. Also the size of the training sample size proved to improve classification ability. Our findings noted that the choice of discretization algorithm does matter in the quest for optimal classification performance. Furthermore, our results indicate that the choice of scoring function

does not affect the classification performance of Bayesian networks. Conclusions drawn from this research are prescriptive particularly for a novice machine learning researcher with valuable recommendations that ensure optimal classification performance of Bayesian networks classifiers.

Keywords

Bayesian networks, Bayesian networks classifiers, Classification, Cloud computing, Denial of Service attacks, Discretization, Probability, Score function

Statement of Original Authorship

I, Lindani Moyo do hereby confirm that all the work contained in this dissertation has not been submitted for any other qualification at this or any other University. Furthermore I confirm that the mini-dissertation does not contain any published information except where due reference has been indicated.

Signature: _____

Date: _____

Plagiarism Declaration

I……student number.....……hereby declare that I am fully aware of the University of Fort Hare's policy on plagiarism and I have taken every precaution to comply with regulations.

Signature:……..;

Publications

Part of the research work presented in this dissertation has been published or has been submitted for publication or review in the following papers:

L. Moyo and K. Sibanda, *Designing an Intelligent Agent for Denial of Service (DoS) Detection in Cloud Computing Environments: A Bayesian Networks Approach*. Poster Presentation, In Proceedings of the 2014 Southern African Telecommunications Networks and Applications Conference, Port Elizabeth, Eastern Cape, South Africa

L. Moyo and K. Sibanda, *On the classification performance of Bayesian networks classifiers in detecting denial of service (DoS) attacks in cloud computing environments*. In Proceedings of the 2015 Southern African Telecommunications Networks and Applications Conference, Arabella, Hermanus, Western Cape, South Africa

Acknowledgements

"To the only wise God our Saviour, be glory and majesty, dominion and power, both now and ever. Amen" (Jude 1:25). I would like to thank my omnipotent Father for his unfailing love, grace and providence that has kept me until this far. Ebenezer!

I am greatly indebted to my research supervisor Professor K. Sibanda for his immeasurable insight, encouragement, guidance and support he offered in this project. A special thanks to the H.o.D of the Department of Computer Science Mr S. Scott for accepting me for the masters programme and providing the necessary equipment to use. Also a heartfelt thanks to Professor M. Thinyane and Mr. Z. Shibeshi for their motivation and concern throughout this study. Thank you to the Masters class of 2015 for the friendship, assistance and knowledge sharing. It was a worthwhile experience.

I would like to acknowledge the on-going support from parents Mr & Mrs C.T. Moyo, my siblings Lungile and Dingani, relatives and Siphephelo, Vukile Ndedwa, Nkosilathi Ndlovu, Gilbert Vundla, Mthulisi Mpofu, Thulani Dube, Noel Nyathi and Mayibongwe Bayana. Their encouragement, love, prayers and support through my academic pursuit was, and still is greatly appreciated.

This work is based on the research undertaken within the Telkom Centre of Excellence in ICT4D supported in part by Telkom SA, Tellabs SA/Coriant, Saab Grintek Technologies, Easttel, Khula Holdings and THRIP. The opinions, findings and conclusions or recommendations expressed here are those of the author and none of the above sponsors accepts any liability whatsoever in this regard.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AODE	Averaged One Dependence Estimators
BD	Bayesian Dirichlet
BDe	Bayesian Dirichilet ("e" for likelihood-equivalence)
BDeu	Bayesian Dirichilet ("u" for uniform joint distribution)
BN	Bayesian Networks
DoS	Denial of Service
GBN	General Bayesian Network
GUI	Graphic User Interface
IaaS	Infrastructure as a Service
IDS	Intrusion Detection System
LL	Log-likelihood
MDL	Minimum Description Length
MIT	Mutual Information Test
NB	Naïve Bayes
NML	Normalized Minimum Likelihood
PaaS	Platform as a Service
SaaS	Software as a Service
TAN	Tree Augmented Naïve Bayes
UCI	UC Irvine
WEKA	Waikato Environment for Knowledge Analysis

Chapter 1: Introduction

This dissertation considers the application of Bayesian Network classifiers to Denial of Service (DoS) attack detection, and attempts to reformulate the task of DoS attack detection in Cloud Computing environments. The purpose of this chapter is twofold, firstly to provide underlying reasons for research described in this dissertation and secondly to present research goals. It serves as an introduction to the research and to the chapters that follow. Section 1.1 outlines the background and motivation for the presented work in this dissertation. The problem statement is provided in Section 1.2 and thereafter the aim and research objectives are discussed in Section 1.3 and 1.4 respectively. Section 1.5 outlines the research questions of this study. The scope of the research is presented and clarified in Section 1.6. The final section is devoted for the description of the dissertation structure and the conclusion of this chapter.

1.1 Background Information

In the past decade cloud computing technology has been a buzzword and has become a major fabric of our contemporary lives. Firstly because of its immeasurable importance to large enterprises that deliver their services through the cloud like Google, Amazon, Microsoft etc and its immediate benefits to the customers or users of cloud services. Secondly cloud computing has been buzzy with activities directed at improving management of computational resources in the cloud. Large enterprises such as Amazon, Sales Force and Google own vast computational resources that need efficient mechanisms put in place to manage and scale up those resources. Scalability is particularly important as cloud computing heavily rely on shared infrastructure. However as more critical infrastructures for example power grid and air traffic control management which are vital to our modern society are migrating to cloud computing environments, security concerns at all levels have paralleled the growth of the computing model. According to recent reports on Internet security, voluminous and sophisticated intrusions targeting the network layer have substantially increased (Citrix, 2014; Alotaibi, 2015). Among the various types of intrusive behaviours, the most security breaches are attributed to Denial of Service (DoS) attacks (Ismail et al., 2013; Vidhya, 2014; Alotaibi, 2015). Increased situational awareness and cyber security solutions in the form of intrusion detection systems (IDS) are required to counter these ever growing sophisticated attacks which threaten cloud computing environments (Eskridge et al., 2009). Cloud network attacks are considered as intrusions and any

endeavour to eliminate the attacks entails first detecting those attacks. ID is a form of security management tool that gathers and analyses information with the aim of identifying possible sets of malicious activities that could compromise the confidentiality, integrity and the availability of computer system resources (Scarfone & Mell, 2007). ID is thus a front-end line of defence for securing computer systems. It is essential for alerting security administrators of impending attacks thus ensuring the smooth and efficient running of cloud-based activities. The development of IDSes is naturally inclined towards incorporating trade-offs in terms of detection accuracy, detection speed etc. Advanced and powerful approaches are required for the ID purpose for efficacy and efficient differentiation of normal activities and malicious activities for early and accurate detection. Securing of cloud computing environments against DoS attacks is realised using ID systems.

To ensure reliability and availability of cloud computing environments it requires early and accurate detection of intrusion attempts. Due to the commercialisation of, the increasing popularity and growing dependence on cloud computing by business enterprises, realization of an attack has devastating repercussions. Securing of the cloud computing infrastructure has therefore become an utmost priority research area for exploration. There are two major approaches to intrusion detection: signature-based and anomaly-based (Modi et al., 2013). In signature detection, the learning algorithm is trained using a dataset in which each instance is labelled as either normal or intrusion. A setback on signature-based approach is that only the modelled attack signatures can be detected. In order for the algorithm to detect novel intrusions that were not present in the training sample, the signature database must be updated by retraining the algorithm with new attack instances. However, in anomaly detection approach the behaviour is modelled and any event that deviates from modelled baseline is flagged as intrusive (Modi et al., 2013). Although this method is able to detect new attack activities its major problem is that it suffers from high rates of false positives which are due to unobserved normal events. In fact most methods that have been designed to detect intrusions have a number of setbacks which are presented later in our literature review. In section 1.2 the research presents the problem analysis.

1.2 Problem statement

The process of intrusion detection usually includes the processes of information gathering identification and classification. Classification is one of the core concepts in the field of pattern

recognition (Duda, Hart & Stork, 1997; Bishop, 2006). It has been applied in numerous problems e.g. document classification, speech recognition, internet search engines etc. Classification is the process of taking an instance of a training sample and predicting the class to which it belongs to based on its features. A classifier is a function therefore used to perform a classification task. A classifier may be used for instance to differentiate between cancerous cells from non-cancerous cells. Classifier models are learned from a given training dataset and subsequently used to infer the class of instances of new observations. Learning of classifier models has been an active research area and numerous researchers have developed classifier models for the intrusion detection problem: neural networks (Ryan, Lin & Mikkulainen, 1998; Reddy & Iaeng, 2013), rule-based (Arjunwadkar & Kulkarni, 2010), support vector machines (Khan, Awad & Thuraisingham, 2007), decision trees (Peddabachigari, Abraham & Thomas, 2004) and Bayesian analysis (Barbará, 2001). However, literature points that Bayesian networks are more advantageous as a classification approach e.g. the graphical representation of the problem domain and the quantification of uncertainty using probability (Uusitalo, 2006).

A preliminary study on the use of Bayesian networks algorithms in the intrusion detection problem is documented by (Barbara, Wu & Jajodia, 2001) and subsequently a number of scholars have applied various Bayesian network models for classification of internet traffic (Scott, 2004; Moore & Zuev, 2005; Bringas & Santos, 2010). Despite the widespread use of Bayesian networks in various fields, the researcher felt there exist uncertainties that need to be explored to yield optimal performance of Bayesian networks and consequently the accurate detection of DoS attacks. Most of the concerns are with regards to the impact of the training sample size, the choice of discretization approach and the score metric used in the induction process on the classification performance (Dougherty, Kohavi & Sahami, 1995; Sordo & Zeng, 2005; Liu, Malone & Yuan, 2012). The aforementioned factors generally tend to affect the predictive power of Bayesian networks, albeit in varying extents commensurate with the factors under investigation. Empirical studies conducted on these factors have been analysed and observed to produce varying results hence this rendered their findings inconclusive (Sordo & Zeng, 2005; Mizianty, Kurgan & Ogiela, 2008; Carvalho, 2009; Bielza & Larrañaga, 2014). Notably, the empirical studies were conducted on synthesized datasets or gold standard hence making it difficult to make the findings to be universal even in real world applications. Consequently, this exposes research gaps in the area of classification capability of Bayesian

networks. Furthermore publications that investigate the connection of structural complexity and the classification performance of Bayesian networks point out that the behaviour of a classifier is not universally ideal in all problem domains, hence it is important to conduct an empirically assessment (Madden, 2009; Bielza & Larrañaga, 2014). This consideration is important with respect to the no-free-lunch theorem. Despite the enhancements or augmentation of model structurally, this does not universally guarantee an improved in performance. Hence this factor is open for investigation. Literature indicates that training sample size is important for learning optimal classifiers (Foody et al., 2006). Inasmuch as sufficiently large sample tends to be representative of the real world domain and crucial in Bayesian models, it is important for learning more accurate probability estimates however it is interesting to note the behaviour with training sample size in Bayesian networks. Statistics researchers have pointed out that training sample size six times the sample dimensionality is adequate. There is uncertainty in how the models behave when inputted with novel examples. Considerations are also on the impact of both model complexity and training sample size on classification performance of Bayesian networks.

Studies also reveal disparities on the effect of the type of discretization approach and scoring function on predictive ability of Bayesian networks (Yang & Chang, 2002; Flores et al., 2011). There are a number of discretization techniques and score metrics to use during the learning process. Therefore there is some uncertainty on what is the best choice of algorithm to incorporate in the design of Bayesian networks (Liu, Malone & Yuan, 2012; García et al., 2013). Flores et al. (2011) describes discretization as a process that strongly impacts Bayesian network classification ability in terms of runtime, bias-variance discretization and accuracy. (Friedman, Geiger & Goldszmidt, 1997; Madden, 2003) describe scoring functions as factors that affect the quality of optimal network structure identified during the induction phase and hence influence the performance of Bayesian networks. Numerous studies have been conducted and no justifications were provided for choosing a particular algorithm while there are other candidate choices. Consequently, this exposes research gaps in the area of classification ability of Bayesian networks.

1.3 Aim

Determine how various factors impact the detection of DoS attacks on the cloud using Bayesian network classifiers. This broader aim has been broken down into specific objectives in Section 1.4.

1.4 Objectives of study

- To review issues of DoS attacks in cloud environments.
- To review Bayesian networks as a method of detection.
- To determine how the complexity of Bayesian network classifiers affects their classification performance.
- To determine how the training set size impacts the classification performance of Bayesian network classifiers.
- To analyse how discretization approaches impact the classification performance of Bayesian network classifiers.
- To determine how the scoring function impact the classification performance of Bayesian network classifiers.

1.5 Research questions

1) How do DoS attacks impact the cloud computing environments?
2) How do Bayesian networks perform classification tasks?
3) How does structural complexity of Bayesian classifiers affect their classification performance?
4) Does the size of the training dataset impact the classification performance of Bayesian classifiers?
5) Does the choice of discretization technique matter and how is the discretization process affected by the training sample size?
6) Does the choice of scoring function result in better classifiers and how are the score metrics affected by the training sample size?

1.6 Scope of the study

Data mining algorithms have been used to model IDS. In this study we used Bayesian networks for this purpose. The Bayesian based IDSes were applied in securing cloud computing

environments. There are a lot of intrusive behaviours that cloud platforms are prone to, however this study is limited to DoS attacks. Also this study does not focus on producing a new classification algorithm or improving the state-of-the art detection of DoS attacks but our study is limited to how the modelling factors of Bayesian networks affect their classification performance.

1.7 Dissertation structure

The dissertation consists of six chapters inclusive of this introduction chapter.

Chapter 1: gives the introduction of the research by presenting the background information to the study. It outlines the underlying reasons for the research work, presents the problem to be investigated and the research scope is also clarified.

Chapter 2: introduces the cloud computing technology from its definition, its essentials characteristics, the service models and the type of clouds. It provides the issues in terms of security that mar the cloud technology and thereafter limits the intrusive behaviour to DoS attacks. A special section is devoted to describe what DoS attacks are and how they are launched.

Chapter 3: provides an in-depth literature review of the concept of Bayesian networks. Furthermore it introduces the Bayesian networks classifiers, which are Bayesian networks adapted for classification tasks. The chapter gives a discussion of the application of Bayesian network classifiers in intrusion detection and outlines also the theory on the induction of Bayesian networks classifiers. It also introduces the classification concept in data mining which is the gist of this study. It thereafter introduces the factors that affect the classification performance of Bayesian networks. There is a relationship between classification performance of Bayesian networks with the type of classifier, the appropriate size of the training sample set, the discretization approach and the score function. This chapter is partitioned into sections that discuss the aforementioned factors and the relevant algorithms under study. After each section a review ensues of the past work done of the factors that impact classification performance. These factors are the focus of the experiments that will be discussed in the later chapters.

Chapter 4: presents the materials used, methodology followed and experiments conducted in this study.

Chapter 5: describes the results and presents the evaluations of the research. The aims of the empirical study are stated, described and the results are analysed. Discussions on the classification performance of BN under the scenarios of the factors investigated.

Chapter 6: presents the conclusions drawn from the research and also give the further areas that can be explored in this particular study.

In addition to the chapters outlined above, **Appendix A-M** gives the snapshots of the output of the Friedman test, Nemenyi test and lastly the Boxplot. **Appendix N** gives the detailed characteristics of the NSL-KDD dataset used as our training dataset.

1.8 Conclusion

The chapter introduced the research study by providing the relevant background information, listing the research objectives and outlined the research questions that will be answered by this study. The chapter stated the research scope and concluded with the dissertation structure. An extensive review of the cloud computing technology and the concept of DoS attacks ensues in Chapter 2.

Chapter 2: Cloud Computing

This chapter introduces the notion of cloud computing and related underlying concepts. Furthermore, it introduces the cloud essential characteristics, the service models and lastly the type of clouds. Presented also are the issues in terms of security that characterise the cloud computing environments. It also discusses the concept of DoS attacks and their detection.

2.1 The notion of Cloud Computing

Computing has undergone a series of major innovations since its inception. Throughout the history of computing, there have been several quests to obviate computer hardware needs from users and detach them from time-shared computing (Zissis & Lekkas, 2012a). It is to that end that we have witnessed the emergence of cloud computing. The cloud technology is the latest innovative and distributed computing paradigm in which dynamically scalable and virtualized resources are accessed as a services over the Internet (Kim & Korea, 2009; Shahzad, 2014). At its simplest state, Cloud computing is basically a means through which computational power, storage, collaboration infrastructure, business processes and applications are consumed as services from anywhere through an internet connection. The cloud technology leverages end-users from their local hardware requirements and in the same vein diminishing overall requirements and complexity on the client-side. Basically this contemporary paradigm seeks to provision computing capabilities (Vaquero et al., 2009). The key capabilities are elaborated on in the ensuing section as essential characteristics.

2.2 Definition of Cloud Computing

The term cloud computing was coined from the cloud symbol that is traditionally used to depict the Internet in flowcharts. Presently, migration to the cloud has reached an advanced stage with end-users storing their personal data such as bookmarks, photographs and their music file etc. on datacenters accessible via the internet. The foundation of the cloud computing technology is virtualization (Zissis & Lekkas, 2012a). Virtualization is a technology that was conceived in the late 1960s. A synopsis of the virtualization technology entails abstraction of the computing resources of services from the underlying operating system (Pearce, Zeadally & Hunt, 2013).

There are a number of definitions that are associated with cloud computing. The web is awash with efforts to pen down a stable definition that grips the essence of the distributed, elastic

computing paradigm. At its broadest level cloud computing is *"A standard IT capability (services, software or infrastructure) delivered via the Internet technologies in a pay-per-use, self-service way"*(Scarfone & Mell, 2007, p. 2; Badger et al., 2011, p. 1).

The definition of cloud computing describes some of the essential characteristics, three service models and four deployment models (Badger et al., 2011; Hogan et al., 2011):

- **Essential Characteristics:-**
    - **On-demand self-service:** end-users can provision computing resources such as network bandwidth and storage on-demand (as needed) in an autonomous mode without human intervention from service providers.
    - **Broad network access:** computing resources are available over networks and can be consumed through standard mechanisms that permit the use of different client platforms e.g. mobile phones, tablets, laptops, and workstations
    - **Resource pooling:** the cloud service provider's cloud resources are pooled to serve end-users using a multi-tenant model that has different physical and virtual resources dynamically assigned and reassigned based on the customer's demand. In a multi-tenant architecture, customers share the underlying infrastructure.
    - **Rapid elasticity:** computing resources can be rapidly scaled up or down at any given time. To the client, the cloud resources are seemingly an infinite pool that can be consumed in any quantity at any time.
    - **Measured service:** the usage of computing capabilities by cloud clients is monitored, controlled and reported by a metering system. The metering system reports the type of service consumed e.g. storage, processing and provides accurate and transparency to both the cloud vendor and client.
- **Cloud Computing Service Models** – In the stead of products, in cloud computing we speak of services and there are basically three categories of service models as illustrated in Figure 2.1. With respect to the OSI model, the physical layer is the first or lowest layer. The first service model layer is known as the Infrastructure as a Service (IaaS).
    - **IaaS** – it is the first and lowest level service available to a cloud computing consumer and provides controlled access to a virtual infrastructure upon which operating systems and application software can be deployed.

- **PaaS** – it is situated above the IaaS. The layer is for deployment of consumer-created applications and also tools provided by the vendor. The vendor exclusively manages or controls the underlying cloud infrastructure such as the network, operating system, servers and storage but however assigns consumers with control rights over the hosted applications and their application hosting platform configurations.
- **SaaS** – it permits the consumer to access the vendor's application hosted on the cloud infrastructure. The hosted applications can be consumed by use of a thin client interface, for instance a web browser. Consumers have no control rights over the underlying infrastructure but are however permitted to alter the application configurations.

- **Types of Cloud Computing** – there exist three types of cloud computing (Furht & Escalante 2010) as depicted in Figure 2-1.

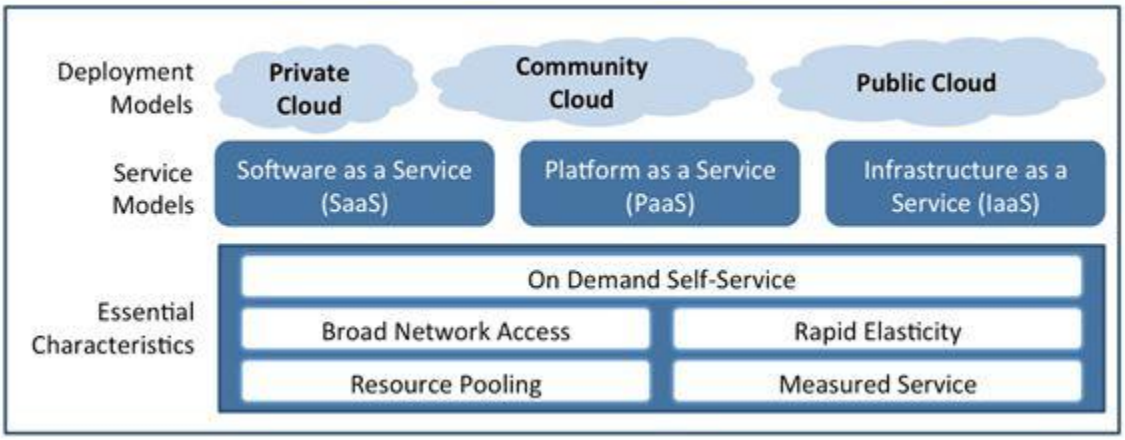


Figure 2-1: Cloud computing model (Shahzad, 2014)

- **Public cloud** - also known as the external cloud. The cloud resources are accessible to the public. Basically, public clouds are generally under the control of a third-party which sells the cloud services.
- **Community cloud** – the cloud deployment is a collaboration of different organizations and is made available to cater for concerns of a specific community. Usually managed by either the involved organizations or a third party and may be deployed on site or otherwise.

- **Private cloud** – refers to a cloud deployed for a private organization. They are basically built for use by exclusively one client, powered with full control over data, security, and quality of service. The cloud infrastructure may be managed or controlled by the organization or a third party. It may be situated on site or off site.
- **Hybrid cloud** – it is generally an environment that is a fusion of public, community, or private cloud models. The clouds each with its unique characteristics are integrated together by a standard that ensures the portability of the data and applications between the clouds.

2.2.1 Issues for Cloud Computing

Typically like any new technology, there exists limitations by which benefits can be achieved and a new approach may come with unique additional challenges. Since cloud computing is in its infancy, still maturing, it has not been spared in this respect. The implementation of cloud computing is thus marred with critical issues. One of the issues is taking critical applications and sensitive data, to public and shared cloud environments. From a consumer's point of view there are a lot of discrepancies on security and trust that are currently inherent in the technology. Many users have mixed feelings about the location and storage of their data.

2.2.1.1 Security in the Cloud

Despite the positive business and technical implications associated with cloud computing, security, trust and privacy are notably the prime hurdles that avert the widespread adoption of the technology (Zissis & Lekkas, 2012a; Khalil, Khreishah & Azeem, 2014). Arguably the most critical security issue in Cloud Computing is detecting malicious activities at the network layer (Subashini & Kavitha, 2011; Zissis & Lekkas, 2012b; Fernandes et al., 2014). Adversaries exploit the existing cloud vulnerabilities and consequently temper with the confidentiality, availability and integrity of cloud resources and offered services. IaaS is the core tier for the other service layers in the cloud. Without a strong protection mechanism embedded in this layer, the security of other cloud layers is compromised. It is to that end that, various IaaS vendors are at task to develop defense schemes that will ensure the protection of their customer's data and applications. Cloud security has thus been widely discussed in industry and academia (Vaquero et al., 2008; Pearce, Zeadally & Hunt, 2013; Sen, 2013). An IDC survey depicted in the Figure 2-

2 below reveals that cloud security is the greatest challenge that is preventing many organizations from migration to the cloud (Ramgovind, Eloff & Smith, 2010).

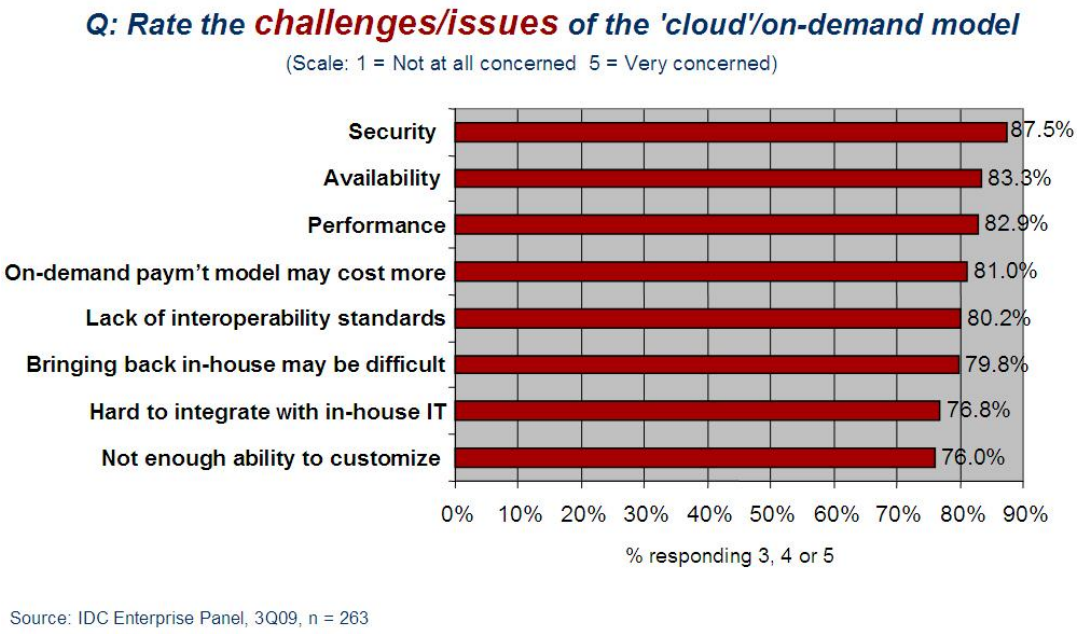


Figure 2-2: Cloud adoption challenges

The Cloud computing environments have brought about certain specific challenges when it comes to the nature of cyber-attacks in the form of theft-of-service (Gruschka & Jensen, 2010), Denial of Service (Khorshed, Ali & Wasimi, 2012; Deshmukh & Devadkar, 2015), cloud malware injection (Zunnurhain & Vrbsky, 2011; Qaisar & Khawaja, 2012), cross VM side channels (Zhang et al., 2012), target shared memory (Khalil, Khreishah & Azeem, 2014) and phishing (Gonzalez et al., 2011).

To date there are several security mechanisms that have been put in place by cloud providers. First and foremost, physical security measures need to be established on-site to protect datacenters from break-ins and other physical violations. Network security uses approaches including firewalls and IPSes which are deployed to analyze network traffic and safeguard data in transit. Deployed also are IDSes to alert network administrators of malicious intrusion attempts (Khalil, Khreishah & Azeem, 2014) and honeypots to detect and thereafter deflect any attempts to unauthorized use of networks.

2.3 Anatomy of Denial of Service attacks

Our contemporary society is largely dependent on the reliability of the Internet, so are the cloud computing environments. It has been proved that it is not adequate to secure communications from eavesdroppers or guard it from being infected with viruses. Traditionally, the security fraternity paid most attention on confidentiality and integrity of information and largely overlooked Denial of Service (DoS) attacks. DoS attacks have become a prevalent security threat that is associated with instability of the internet (Ismail et al., 2013).

2.3.1 What is a Denial of Service?

Denial of Service (DoS) is a form of attack that disrupts a legitimate activity and makes sure that information or data is not available to its intended users (Specht & Lee, 2004; Abliz, 2011). The attack paralyses normal services and consequently affecting the usability and reachability of network services such as web, mail, voice and data. DoS attacks tend to tarnish the reputation of the service provider. There are several methods that are used to implement this attack. A DoS attack is executed by sending useless messages to a target host so as to interfere with its operations, thus making it to crash or process useless operations. The attack renders a machine or a network resource to be unavailable or to extremely slow for use by legitimate users (Mirkovic, 2003).There are however other ways with which services are made to be inaccessible rather than just sending an avalanche of IP packets. This involves the exploitation and attacking of various loopholes that exist on the victim to make it unstable.

2.3.2 DoS attacks mechanisms

There are basically two ways which are used to interfere with a legitimate operation (Douligeris & Mitrokotsa, 2004):-

a) Exploitation of vulnerability inherent on a target machine or application - this entails sending messages made in a specific manner to take advantage of a given vulnerability.
b) Sending vast number of messages so as to consume some key resource – useless traffic is sent to a key resource such as CPU time, bandwidth, and memory at the target machine. It is therefore that target applications, host or network spends critical resources processing malicious traffic and not attending legitimate clients.

2.3.3 Network Flooding

To prevent the exploitation of a vulnerability, simply patching of the system vulnerability is required. It is however difficult to manage flooding-based DoS attacks. Presently, it is easy to launch flooding-based attacks on the Internet because attack tools such as hping and nmap are readily available and easy to use. Basically any Internet user can use them with great ease. Discussed briefly below are common flooding-based DoS attacks:

- UDP flood entails sending an overwhelming number of UDP packets with spoofed source IP addresses to either random or specified ports on the victim machine (Specht & Lee, 2004). The spoofing of the attacking packets helps to hide the identity of the secondary victims due to the fact that the return packets from the victim host will be sent to the spoofed addresses.
- ICMP flood follows a similar mechanism and depicts a behavior similar to that of the UDP flood (Bogdanoski & Risteski, 2011). Basically, large volumes of ICMP_ECHO_REPLY packets are generated and directed to a victim machine. The essence of the attack is to ensure that the number of ICMP_ECHO packets significantly saturates the CPU and network resources such as bandwidth of the victim network connection to make full responses. Smurf attack is a well-known instance of ICMP flood (Bogdanoski & Risteski, 2011).
- Lastly, TCP flood is a form of DoS attacks that exploits the vulnerability inherent in the implementation of the TCP protocol. The most prevalent instance of TCP flood is the TCP SYN flood. It takes advantage of the vulnerability in TCP's three-way handshake process and its inability in maintaining half-open connections (Divakaran, Murthy & Gonsalves, 2006). Figure 2-3 shows the diagram of a three-way handshake.

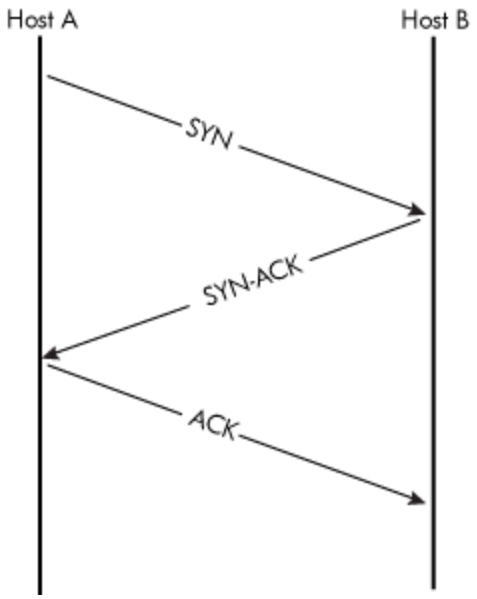


Figure 2-3: TCP three-way handshake (Divakaran, Murthy & Gonsalves, 2006)

The mechanism of the three-way handshake as per TCP connection is described in-depth below.

a) The client initiates the connection by sending a TCP-SYN packet to the server to request a service.
b) Upon reception of the SYN packet, the server enters into a SYN-RECEIVED mode and allocates a Transmission Control Block (TCB) to store the information about this connection and subsequently responds to the connection request with a SYN-ACK packet.
c) Lastly, in normal and legit conditions, the client sends an ACK upon reception of the SYN-ACK packet from the server and thus finalizes the establishment of the connection with the server.

2.4 Detection method

This section aims to deliver an overview of the security schemes that are designed for the detection of DoS attacks. For the purpose of detecting DoS attacks, detection can be achieved through the use of two complementary strategies namely (Butun, Morgera & Sankar, 2013):

- signature-based
- anomaly-based

2.4.1 Signature-based detection

It is also known as misuse-based detection. Its quest is to detect an attack by defining or modeling a set of rules or signatures or predefined profile baselines that can be used to compare with a given behavior to decide whether it is intrusive (Paxson, 1999; Modi et al., 2013). This approach is characterized by high detection rates and low number of false positives. A slight deviation or variation in the intrusive behavior may however by-pass the detection system. It is thus to that end that signature-based engines are an efficient solution for the detection of known attacks but they have no capacity to detect unknown attacks. The approach requires constant maintenance and updating of signature engines.

2.4.2 Anomaly-based detection

Anomaly-based detection is basically concerned with identifying behaviours or events that might be anomalous with respect to normal behavior (Modi et al., 2013). The approach is based on the assumption that intrusive patterns differ mathematically or statistically with normal behaviours. The classifiers used to detect the anomalous activities are modeled from advanced mathematical or statistical techniques. For example, normal network traffic can be collected over a period of normal operations and thereafter a modeling technique is applied to build the normal network profiles. During the detection phase, any degree of deviation between the current network flow and the modeled normal network profiled is calculated. In the event that the deviation overshoots past a pre-set threshold, the current network flow will be flagged as an intrusive pattern. Anomaly detection systems have the advantage that they can detect previously unknown events, zero-day attacks.

2.4.3 DoS detection in cloud computing

According to CSA (2010), DoS attacks top the list of threats to cloud computing environments. Worldwide Infrastructure Security Reports reveal that 94% of cloud computing clients have been subject to security attacks. Notably, 76% of the attacks are DoS related attacks (Deshmukh & Devadkar, 2015). About 43% experienced partial or total infrastructure outages due to DoS related attacks (Deshmukh & Devadkar, 2015). Many prominent websites such as Yahoo, Facebook and Twitter have been affected by DoS attacks. Not only have DoS attacks become more prevalent, they have intensified in destructive capacity. Numerous detection methods have been constructed to counter DoS attacks in cloud platforms. There exist many solutions towards

DoS traffic detection and each based on differing ideas. Below are synoptic reviews of publications of techniques used to detect DoS attacks on the cloud.

Lonea, Popescu & Tianfield (2013) applied the Dempster-Shafer theory based IDS to detect DoS attacks on the cloud. Their solution proved to be efficient in reducing false alarm rates in their detection. (Ismail et al., 2013) implemented a cloud-IDS based on Covariance Matrix approach to detect DoS attacks. The algorithm was used to model a traffic profile for normal activities. Their results indicate that the approach is quite effective for detecting abnormal activities. Navaz, Sangeetha & Prabhadevi (2013) constructed an entropy based anomaly detection system for detecting DoS attacks on the cloud. Artificial Neural Networks (ANNs) are a machine learning technique inspired by the observation from the human nervous system with an interconnected web of neurons (Moyo & Sibanda, 2015). ANN have been successfully applied in the detection of DoS attacks, such as the work presented by (Alfantookh, 2006). The authors of (Ahmad et al., 2009) conducted a study on the use of multiple layered perceptron architecture with resilient back propagation for modeling a detection system. In comparison with other neural network approaches, the resultant tool has better performance with more accuracy and precision on the detection rate of DoS attacks.

2.5 Conclusion

This chapter gave an introduction to the cloud computing technology. The underlying concepts were stated and a comprehensive definition was provided. The essential characteristics, service models and the four deployments of the cloud were outlined. This chapter also gave a review of issues associated with cloud computing technology. Security challenges and the vendor security mechanisms were discussed in this chapter. Finally the chapter ended by discussing the anatomy of DoS attacks and how the attacks are launched. Furthermore, DoS detection on the cloud was briefly discussed. The information presented is essential in order to appreciate and understand the case-study area of this study. Chapter 3 will deal with the theoretical background and literature of the fundamental of Bayesian networks.

Chapter 3: Bayesian Networks Theory

The focal point of this study revolves around the concept of Bayesian networks. Bayesian networks are a member of the graphical models family and they are used to capture domain knowledge for reasoning and decision making in environments fraught with uncertainty. Uncertainty arises as a result of vagueness and missing or incomplete knowledge in stochastic environments. The probability theory coupled with the statistical analysis in Bayesian networks offer an antidote to the problem of uncertainty hence, making it possible to draw conclusions. Probability is the underlying core of Bayesian network theory and thus a synoptic review of the probability theory will precede the in-depth introduction to Bayesian networks. The inductive learning of accurate Bayesian network classifiers is an important goal in the field of machine learning. Several factors have been noted to adversely impact the classification performance hence the primary focus has been drawn to investigating the influences on the accuracy of BN classifiers. In this chapter we review literature of the existing research on the classification accuracy of Bayesian network classifiers. Much attention will be on the factors outlined in the objectives by which the research work is based. Incredible interest is on the model complexity, type of discretization approach, on sample size and lastly the type of scoring function used. We carefully review previous work done by academia and industry in various case studies. The aim is to identify the approaches used in each case study. The chapter is outlined as follows. The primary concepts of the probability theorem are briefly discussed in Section 3.1. The definition of Bayesian networks based on literature will be provided in Section 3.2. Section 3.3, the notation of Bayesian networks which is of interested in this dissertation will be given. Section 3.4 addresses the foundation of Bayesian networks and the generic flow of information in Bayesian networks. The various kinds of probabilistic inference and variable elimination algorithm are discussed in Section 3.5. Bayesian parameter and structure learning are in discussed in Section 3.6.

3.1 Probability theory

Mathematics provides the foundation of the probability theory. It was conceived more than two thousand years ago and has been a foundational block of all sciences. Probability theory is highly recommended for empirical studies fraught with uncertainty. It provides a rational framework for making intelligent decisions in uncertain environments. Heckerman (1996) states that probability

is the degree of belief in an event and it can be expressed on a linear scale with a range from 0 to 1. There are two schools of thought in interpreting probability: frequentism and Bayesianism. Frequentists interpret probability as a measure of the frequency of trials while Bayesians view probability as a subjective measure of the outcome of a trial. The probability theory is one of the core components of Bayesian networks. Before we delve into the theory of Bayesian networks, it is imperative to commence by reviewing the underlying principle which is the Bayes' theorem in the ensuing section.

3.1.1 Bayes' theorem

For a better understanding of the Bayesian networks, it is a prerequisite to familiarize with some notions about the Bayesian theory. The Bayesian theory dates back from the eighteenth century and this work is attributed to the English statistician Reverend Thomas Bayes.

The Bayes theorem is

$$\boldsymbol{P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}} \qquad (\mathbf{3.1})$$

where

$P(X)$ is the prior probability or marginal probability of X.

$P(X|Y)$ is the conditional probability of X given Y. It is also known as the posterior probability because it is dependent on the specified value of Y.

$PY|X)$ is the conditional probability of Y given X.

$P(Y)$ is the prior or marginal probability of Y, acts as a normalizing constant.

Intuitively, the Bayes' theorem is a description of an observer's beliefs of an event X is updated by the occurrence of event Y. Bayes' theorem is used to compute "inverse probability". This means that if the conditional probability $P(X|Y)$ is known, consequently the probability of $P(Y|X)$ can be computed. The formula basically all entails an overall summary of the theory behind Bayesian tool internals.

3.2 Bayesian Networks

After the discussion on the Bayesian theory, this section is devoted to explaining the term; *Bayesian networks*. The question is posed, *"What are Bayesian networks?".* Literature is awash with differing definitions on what the term Bayesian networks refers and notably all the formulations amount to the same thing. Pearl (1993) coined the term "Bayesian Networks" in 1988. He referred Bayesian networks as (Pearl, 1993, p. 52):

"A probabilistic network (also referred to as a belief network, Bayesian network or causal network) consists of a graphical structure, encoding a domain's variables and the qualitative relationships between them, and a quantitative part, encoding probabilities over the variables".

The above statement is one of the simple, used and most cited definitions of a Bayesian network. In this dissertation, the term Bayesian networks will stand in for all the terms such as causal networks, belief networks that are interchanged in most literature articles.

Heckerman (1996, p. 1) introduced a simpler, shorter definition of the Bayesian network which he stated as follows:

"Bayesian network is a graphical model for probabilistic relationships among a set of variables."

Korb & Nicholson (2004, p. 26) proposed a more recent definition of Bayesian network which he stated as follows:

"A Bayesian network is a graphical structure that allows us to represent and reason about an uncertain domain."

Korb & Nicholson (2004)'s definition provides a broader definition that introduced the term reasoning which adds a new dimension to the concept of Bayesian networks. Basically this implies that new observations will allow updating of knowledge and the belief embedded in a Bayesian network thus Bayesian networks can therefore be used for reasoning in environment marred with uncertainty based on probabilities.

Bayesian algorithms offer the following advantages for data analysis especially when coupled with statistical techniques (Heckerman, 1996, 1997):

- They are readily capable of handling scenarios where there are missing data entries because the model can encode dependencies among all the modelled variables. For instance take into account a classification problem where two input variables are strongly anti-correlated. The correlation is not a challenge for standard supervised learning techniques, provided all input are measured in every case. In the event that one of the inputs is not measured, most models tend to produce an inaccurate prediction because they do not encode the correlation between the input variables. Bayesian networks, however offer a natural way to encode such dependencies.
- They allow the learning of causal relationships and thus providing an effective way to gain understanding of the problem domain and consequently allow predictions to be made on the presence of new evidence.
- When coupled with Bayesian statistics, they present an ideal framework that combines domain knowledge and data because the model is embedded with both causal and probabilistic semantics. Bayesian networks have a causal semantics that makes the encoding of causal prior knowledge fairly straightforward. Additionally, Bayesian networks quantize the strength of causal relationship using probabilities.
- Bayesian statistical methods coupled with Bayesian networks impart efficacy and a principled approach to evading the over-fitting of data.

The following section introduces Bayesian networks by an extensive discussion from the first principles.

3.3 Notation

Bayesian networks are a formalism that provides a powerful framework for modelling relationships among random variables in domains fraught with uncertainty. They provide a representation of joint probability distribution on the modelled variables. Bayesian networks are a dual-concept model that consists of the qualitative component where features from the graph theory are implemented and an associated component which comprises of real-valued functions over all the modelled variables on the graph. A Bayesian network built for a set of variables $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$ has the following properties (Heckerman, 1996):

1. A network structure $\boldsymbol{S}$ that encodes a set of conditional independence assertions about n random variables in $\mathbf{X}$.
2. Each variable has an associated local probability distribution $\boldsymbol{P}$.

The abovementioned properties define the joint probability distribution for $\mathbf{X}$. The structure $\boldsymbol{S}$ is a directed acyclic graph. The variables in $\boldsymbol{S}$ are in a bijection function with the variables $\mathbf{X}$. The term $\boldsymbol{X_i}$ is used to denote a variable and its corresponding variable. $\mathbf{Pa}_{\boldsymbol{i}}$ denotes the parents of the variable $\boldsymbol{X_i}$ in $\boldsymbol{S}$ as well as the corresponding variables of those parents. A lack thereof of an arc in structure $\boldsymbol{S}$ encodes conditional independencies. The probability distribution for $\mathbf{X}$ given structure $\boldsymbol{S}$ is given by

$$\boldsymbol{p}(\mathbf{x}) = \prod_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{p}(\boldsymbol{x_i} \,|\mathbf{pa}_{\boldsymbol{i}}) \qquad (\mathbf{3.2})$$

Equation 3.2 represents the local probability distributions $\boldsymbol{P}$ which is a product of individual variable distributions. Together the structure $\boldsymbol{S}$ and the probability distribution $\boldsymbol{P}$ encode the joint probability distribution $\boldsymbol{p}(\mathbf{x})$.

In a Bayesian network, the probabilities encoded maybe either Bayesian or physical. If a Bayesian network is constructed solely from prior knowledge, the probabilities encoded will be Bayesian. When a network is built from data, the structure will be encoded with physical probabilities. The focus of this research we focus on learning a Bayesian network and its corresponding probabilities from data. In subsequent sub-section, we present a discussion on constructing a network from data.

In building a Bayesian network, a directed acyclic graph is constructed that encodes the conditional independence assertions. Using the chain rule of probability:

$$\boldsymbol{p}(\mathbf{x}) = \prod_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{p}(\boldsymbol{x_i}|\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_{i-1}}) \qquad (\mathbf{3.3})$$

This basically means that for every $\boldsymbol{X_i}$, there is a corresponding subset $\boldsymbol{\pi_i} \subseteq \{\boldsymbol{X_1}, \dots, \boldsymbol{X_{i-1}}\}$ such that $\boldsymbol{X_i}$ and $\{\boldsymbol{X_1}, \boldsymbol{X_2}, \dots, \boldsymbol{X_{i-1}}\} \backslash \pi_{\boldsymbol{i}}$ are conditionally independent given π_i. Therefore for any **:**

$$\boldsymbol{p}(\boldsymbol{x_i} \mid \boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_{i-1}}) = \boldsymbol{p}(\boldsymbol{x_i}|\, \boldsymbol{\pi_i}) \qquad (\mathbf{3.4})$$

Equation 3.3 and 3.4 combined will result in

$$\boldsymbol{p}(\mathbf{x}) = \prod_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{p}(\boldsymbol{x_i}|\, \boldsymbol{\pi_i}) \qquad (\mathbf{3.5})$$

A comparison of Equation 3.3 and 3.4, we observe that the variable set $(\boldsymbol{\pi_1}, \dots, \boldsymbol{\pi_n})$ corresponds to the Bayesian network parents $\boldsymbol{(Pa_1, \dots, Pa_n)}$. This will specify the edges linking the variables in structure $\boldsymbol{S}$.

According to Heckerman (1997), the structure $\boldsymbol{S}$ is dependent on variable ordering and the variables of interest $i = 1, \dots, n$. The approach suffers from a major challenge in that if the variable ordering is chosen without care, the resultant network structure may specify less conditional independencies among the variables. There are $\boldsymbol{n}!$ possible variable orderings to explore and seek for the best one. Beside the variable ordering approach for constructing a Bayesian network, there exists another approach that derives a network from observations. Experts will model a Bayesian network for a given set of variables by simply drawing edges connecting variables in a cause and effect relationship. The final phase in constructing a Bayesian network involves encoding the associated local probability distributions $\boldsymbol{p}(\boldsymbol{x_i} \mid \mathbf{pa}_i)$.

The arc-direction in Bayesian networks represents causal dependency between variables. Events in a domain are represented as variables. The events are connected by directed edges. An edge is a representation of the causal relationship between the events and the direction of edge is from the cause to the effect. In Bayesian networks, there exists a parent-child relationship if there is an edge from the former to the later. The degree of the strength of the parent-child connections are quantitatively determined by use of probability values. These values are automatically updated as new information is observed.

3.4 Conditional Probabilities

Bayesian networks are a formalism capable for reasoning under uncertainty by use of a graph structure that will present the relations between events. Bayesian networks handle uncertainty in its structure through the use of conditional probabilities.

A variable X with n states $x_1, x_2, \dots, x_n$, the probability distribution $P(X)$ over these states is represented as:

$$\boldsymbol{P(X) = (x_1, x_2, \dots, x_n);\ x_i \geq 0;\ \sum_{i=1}^{n} x_i = 1} \qquad (\mathbf{3.6})$$

where x_i represents the probability of X being in a state.

The fundamental rule of probability

$$\boldsymbol{P(x|y)P(y) = P(x, y)} \qquad (\mathbf{3.7})$$

where $P(x, y)$ is the probability of the event x and y. Furthermore, $P(x|y)P(y) = P(y|x)P(x)$ this consequently leads to the Bayes' rule:

$$\boldsymbol{P(x, y) = \frac{P(x|y)P(y)}{P(x)}} \qquad (\mathbf{3.8})$$

The probabilities are elicited from a domain expert (Stefanini, 2008; Paulson et al., 2011) or can be induced from a from dataset (Cooper & Herskovits, 1992a; Heckerman, 1996).

Bayesian networks provide a concise and composite representation of a domain and its uncertainty. Any subset of the modeled variables can be conditioned upon reception of new information or evidence. Bayesian networks support the "flow of information", reasoning, in any direction. During the inference process, any variable may be a query variable and can also be an evidence variable as well. In a probabilistic network, the model will compute the posterior probability for the query variables given the evidence variables.

3.5 Bayesian Networks as classifiers

Bayesian network classifiers are a special kind of Bayesian networks tailored for classification tasks (Friedman, Geiger & Goldszmidt, 1997). With Bayesian network classifiers, the conditional probabilities for node are a quantization of the attribute dependencies given their parents. A feature which consists of attributes $\{\boldsymbol{X_1, X_2, X_3, \dots, X_n}\}$ is represented as variables in a Bayesian network and $\boldsymbol{(x_1, x_2, x_3, \dots, x_n)}$ are the attribute values of an instance $\boldsymbol{E_i}$. The classification node C is always the top node in a Bayesian network. Let the value of C for instance E be c. According to equation 3.9, a Bayesian network classifier is defined as

$$\boldsymbol{c(E) = \underset{c \epsilon C}{\operatorname{argmax}} P(c)P(x_1, x_2, \dots, x_n|c)} \qquad (\mathbf{3.9})$$

3.5.1 Naïve Bayes Classifiers

The simplest type of Bayesian network classifier is the Naïve Bayes (NB) classifier (Langley, Iba & Thompson, 1992; Friedman, Geiger & Goldszmidt, 1997) (See Figure 3-3) since conditional independence given the class is assumed amongst modelled variables. In NB, there exist connections only from the parent node to the child nodes. No connections are permitted between any pair of child nodes and this concept is prominently referred to as the conditional independence assumption. By conditional independence we are referring to independence probabilistically; X is independent of Y given C whenever $P(X|Y,C) = P(X|C)$ for all possible values of X, Y and C, whenever $P(C) > 0$. Due to its simple structure, building a NB is done with ease and it does not require a structure learning algorithm. Thus,

$$\boldsymbol{p(c|x) \propto p(c) \prod_{i=1}^{n} p(x_i|c)} \qquad (\mathbf{3.10})$$

NB classifies using maximum a posterior (MAP) hypothesis by selecting:

$$\boldsymbol{\underset{c_i}{argmax}\ P(c_i) \prod_{j=1}^{n} P(x_i|c_i)} \qquad (\mathbf{3.11})$$

The independent assumption accounts for the simple and computational efficiency of the NB model. Despite that the attribute independence assumption being grossly violated in the real-world, literature Langley, Iba & Thompson (1992) reveals that NB performs remarkably well across various domains.

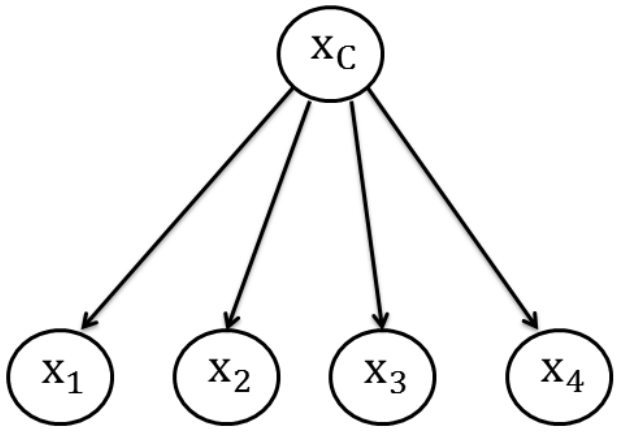


Figure 3-1: NB

NB classifier learns from the training data, the conditional probability of each variable $\boldsymbol{X_i}$ given the class variable *C*. Any classification is performed through the application of the Bayes' theorem which computes the probability of C given the particular instance of $\boldsymbol{X_1, X_2, \ldots, X_n}$, and subsequently predicts the class with the highest posterior probability using:

$$\boldsymbol{P(C_i|X) = \frac{P(C_i)P(X|C_i)}{P(X)} = \frac{P(C_i)\ \prod_{j=1}^{N} P(x_j|C_j)}{\sum_{K}^{k=1} P(C_k) \prod_{j=1}^{N} P(x_i|C_k)}} \qquad (\mathbf{3.12})$$

where K is the number of classes, J is the number of variables and $P(x_j|C_k)$ is the calculated conditional probability of an observed event's variable x_j given the class C_k. With respect to the conditional independence assumption, conditional probabilities are multiplied and this results in a reduced less expensive learning process. After the computation of $P(C_i|X)$ for each class, the induction algorithm assigns an instance to the class with the highest probability.

At training time, NB algorithm generates a one-dimension with a time complexity $O(tn)$, where t is the number of training examples. The space complexity is $O(cnv)$ where v is the average number of values per attribute and c is the number of classes. The resulting time complexity at classification time is $O(cn)$, while the space complexity is $O(cnv)$.

3.5.2 Tree Augmented Naïve Bayes Classifier

The supposition of conditional independence among attributes given the class in the training set does not hold true in real life domains. However, the classifier performs outstandingly in practice even in datasets with strong attribute correlations. Friedman, Geiger & Goldszmidt (1997) introduced the tree augmented Naïve Bayes classifier (TAN) as a natural enhancement of the NB classifier (Friedman, Geiger & Goldszmidt, 1997) (See Figure 3-2). TAN classifier is based on a NB classifier structure and the restricted family of Bayesian networks in which the classification variable has no parents and each of the child attribute has as parents the class attribute and at most one other attribute. As depicted in the Figure 3-2, there exists a edge in the structure from X_i to X_j. This implies that the conditional independence in TAN has been relaxed been attribute X_i to X_j. Therefore the attribute X_i influences the class probabilities of attribute X_j. It is that end that the posterior probability $P(C|X_1, \ldots, X_n)$ takes into account all the involved variables. Edges among the attributes are taken into consideration as to account for the correlations among the attributes. In a structure of n attributes, $n-1$ edges are added to relax the independence

assumption (Friedman & Goldszmidt, 1996). The learning procedure of the TAN classifier is described in-depth in Algorithm I adapted from (Flores et al., 2011).

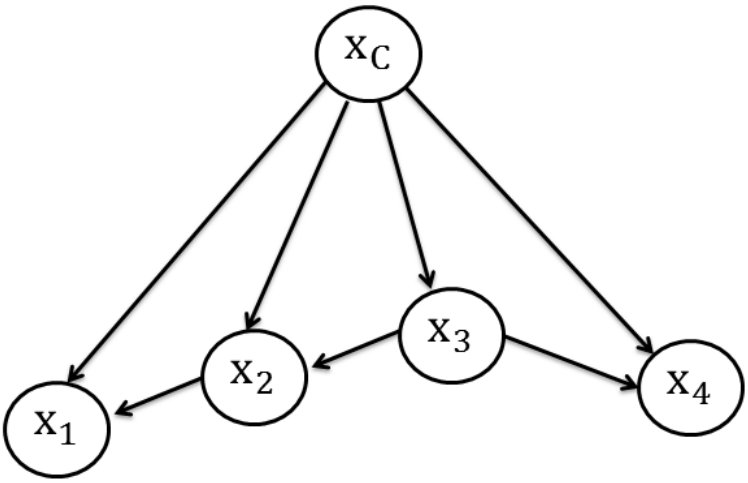


Figure 3-2: TAN

Algorithm I The TAN algorithm

1: Calculate $\boldsymbol{MI}(\boldsymbol{A_i}, \boldsymbol{A_j}; \boldsymbol{C})$ for each pair of attributes ($\boldsymbol{i < j}$ where $\boldsymbol{i, j} = 1 \ldots \boldsymbol{n}$)
2: Build a complete undirected graph (UG) with all the attributes
3: ∀ arc connecting $\boldsymbol{A_i}$ and $\boldsymbol{A_j}$assign the weight $\boldsymbol{MI}(\boldsymbol{A_i}, \boldsymbol{A_j}; \boldsymbol{C})$
4:{**CHOW AND LIU ALGORITHM: Maximum Weight Spanning Tree**}
5: Add the two arcs with maximum weight
6: **while** (arc_in_BN ≤ (n – 1)) **do**
7: add the arc with maximum weight if not cycle in BN
8: **end while**
9: Transform the UG into a directed graph (DG) selecting a root
10: Complete the DG adding C and arcs from C to $\boldsymbol{A_i}$, $\forall_{\boldsymbol{i}}$
11: Compute the CPTs

It is considered a fair trade-off between model complexity and model accuracy. At training time TAN generates a three dimensional table with a space complexity $O(c(nv)^2)$. The time complexity of forming the three-dimensional probability table is $O(tn^2)$ and $O(cn^2v^2)$of creating the parent function. A maximum spanning tree is then generated with time complexity $O(n^2 \log n)$. At classification time TAN only needs to store the probability tables with space complexity $O(cnv^2)$. The time complexity of classifying of a single example is $O(cn)$.

3.5.3 Averaged One Dependence Estimators (AODE)

AODE is regarded as an improvement of NB (Webb, Boughton & Wang, 2005). As stated previously the conditional independence assumption in NB is highly violated in the real datasets and this consequently leads to Equation 3.10 incurring significantly high error rates. Lazy Bayesian Rules (LBR) (Zheng & Webb, 2000) and Super-Parent TAN (SP-TAN) (Keogh & Pazzani, 1999) are some of the resultant approaches in the quest of obviating the error problem. They both rely on relaxing the conditional independence assumption in NB. In-depth discussions of LBR and SP-TAN are found in their respective publications cited. Webb, Boughton & Wang (2005) states that although the training of LBR and SP-TAN is associated with low error rates, their computation are undesirably expensive. This is mainly attributed to (1) model selection (2) probability estimation. AODE is an enhancement of both LBR and SP-TAN approaches. It achieves its efficiency by restricting its search space to 1-dependence classifiers only. This thus obviates the model selection problem. To this end, computation is greatly minimised. However, 1-dependence classifiers require that each variable to have one other variable as parent and the specific attribute must be selected. To solve this problem, a limited number of 1-dependence classifiers are selected and aggregated. In principle, AODE exclusively accommodates models characterized by pair-wise attribute interdependencies. The model is therefore regarded as an ensemble of all dual parent models with one of the attributes as one of them (see Figure 3-3).

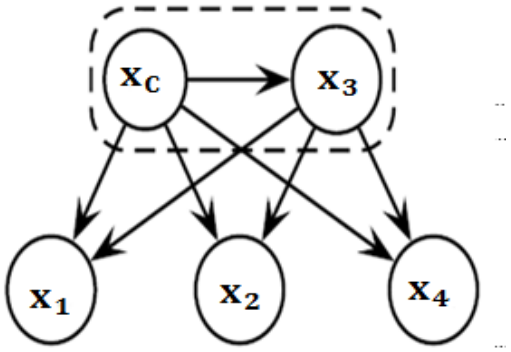


Figure 3-3: AODE

Given a set of random variables $\boldsymbol{X_1, X_2, \ldots, X_n}$, to estimate the probability of C we use the formula:

$$\boldsymbol{P(C|x_1, \ldots, x_n) = \frac{\sum_{i:1\leq i\leq n \wedge F(x_i)\geq m} P(c,x_i) \prod_{j=1}^{n} P(x_j|c,x_i)}{\sum_{c'\in C} \sum_{i:1\leq i\leq n \wedge F(x_i)\geq m} P(c',x_i) \prod_{j=1}^{n} P(x_j|c',x_i)}} \quad \textbf{(3.13)}$$

At training time, AODE has a $O(tn^2)$ time complexity, whereas the space complexity is $O(k(nv)^2)$. The resulting time complexity at classification time is $O(kn^2)$, while the time space complexity is $O(k(nv)^2)$.

3.5.4 General Bayesian Networks

Discussed in previous sections are Bayesian network classifiers belonging to the restricted family. Generally, the maximum number of parents that a node is designed to have is relatively low. Their main difference from other Bayesian classification models is in the manner in which P($x_1|x_1, x_2,\ldots, x_n$,C) is computed. The main advantage of this family of semi-Naïve Bayesian classifiers is the convenient trade-off between accuracy and computational time. With GBN classifiers, an exhaustive search for the network structure using inductive learning algorithms such as hill climbing which performs arc addition, deletion and reversal (Buntine, 1996). GBN classifier models are learnt when the maximum number of parents is greater than 2. A visual notable feature with GBN classifiers is that the classification node x_c is just like other random variables $x_1, \ldots., x_n$ (Madden, 2009) (see Figure 3.4).

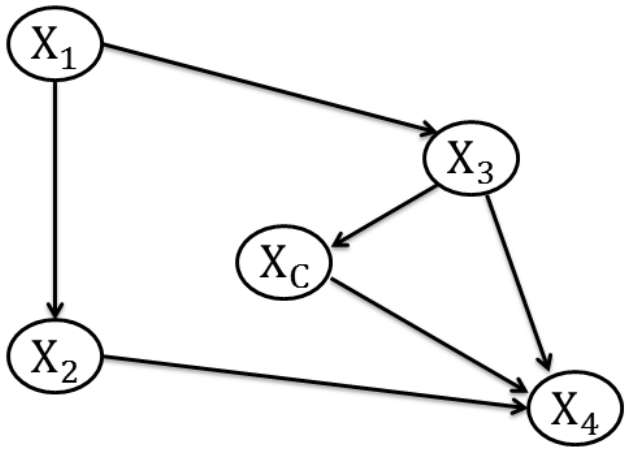


Figure 3-4: GBN

The learning procedure associated with GBN adapted from (Cheng & Greiner, 1999) is outlined in Algorithm II.

Algorithm II The GBN Algorithm

1: Take the training set and the feature set (along with the node ordering) as input
2: Call the (unmodified) CBL1 algorithm
3: Find the Markov blanket of the classification node
4: Delete all the nodes that are outside the Markov blanket
5: Learn the parameters and output the GBN

3.6 Learning Bayesian Networks

Induction of Bayesian networks from data is an attractive field of research that has attracted a lot of research over the last decade (Heckerman, Geiger & Chickering, 1994; Peña, Björkegren & Tegnér, 2005). Learning of Bayesian network is two-fold:

1. Learning the graphical structure
2. Parameter learning

It is rather a trivial exercise to learn the parameters that are optimal for a given graphical structure given the training dataset. The learning algorithm makes use of the empirical conditional frequencies embedded in the dataset. Roure (2004, p. 19) formally defines learning in the context of Bayesian networks as follows: "*Given a data set, infer the topology for the belief network that may have generated the dataset together with the corresponding uncertainty distribution."*.

From this definition we can simply infer that the learning of a Bayesian network is the extraction of a model by a learning algorithm from a given the dataset.

3.6.1.1 Structure Learning

The learning of the structure is known to be NP-hard (Buntine, 1996). There are generally two approaches for learning Bayesian network structures from data. The first method is based on the analysis of dependence relations that considers learning as a constraint problem (Friedman, Geiger & Goldszmidt, 1997). Inductive algorithms used in this approach work by testing for conditional independencies and dependencies in the dataset and then attempt to find a network structure that best exhibits these dependencies and independencies. However this approach has limitations in that they are highly prone to failure during the statistical tests. Consequently if an incorrect result is produced, this will thus lead to learning of an inaccurate network structure.

The second approach is the search and scoring based method and specified as a statistical model, therefore considers the learning process as model selection problem. Prior the learning process, the algorithms define a score that searches for a candidate structure that best fits the observed data. Example of commonly used scores metrics include Minimum Description Length (MDL) (Suzuki, 1993) and Bayesian score (Cooper & Herskovits, 1992a; Heckerman, 1996). For an extensive discussion, the reader is advised to refer to Section 3.10.

3.6.1.1.1 Learning strategies

The number of network structures produced grows super-exponentially in the number of variables. It is to that end that learning algorithms follow a greedy search path. Hill-climb algorithm is an optimization method that belongs to the local search family (Selman & Gomes, 2002). It iteratively solves a problem by initially starting with an arbitrary node-ordering and thereafter incrementally changing the node-ordering in search of the optimal node ordering. Resultant node-ordering is evaluated using a scoring function, then selects the best node ordering. The process iterates until the addition of a new node doesn't increment the score of the resulting node-ordering.

One of the most prominent and classical structure learning algorithms is the K2 algorithm (Cooper & Herskovits, 1992b). K2 algorithm is based on scoring search and is a fusion of the Bayesian scoring metric and hill climbing algorithm. The K2 algorithm uses a greedy search (Lerner & Malka, 2011). To use the K2 algorithm, node ordering must be determined initially as well as the maximum number of parents of each node. This consequently accounts for its computational efficiency because the search space would have been significantly reduced. However, an improper node ordering is said to lead to poor performance. Thus for improved performance, it is crucial to provide the K2 algorithm with an optimum node ordering.

3.6.1.2 Parameter learning

After the Bayesian network structure has been learnt, it is necessary to represent the joint probability distribution which will allow for probabilistic inference. However, it remains a challenge to specify the quantitative component of Bayesian networks which involves filling the Conditional Probability Tables (CPTs). There are basically three ways to populate the CPTs:

- Elicitation of the probability values from domain experts. However, this tends to be subjective, time consuming and expensive.
- Estimating the probabilities from domain data.
- A combination of both domain knowledge and prior data.

The Expectation-Maximization (EM) algorithm is one of the algorithms that have been studied in learning parameters for Bayesian networks from data (Heckerman, 1996).

3.7 Application of Bayesian network classifiers in the Intrusion detection problem

Table 3-1: Research on Bayesian classifiers and their application to intrusion detection domain

BN classifier	Origin of model	Application to intrusion detection scenario
NB	(Langley, Iba & Thompson, 1992)	(Barbara, Wu & Jajodia, 2001) (Panda & Patra, 2007)
TAN	(Friedman, Geiger & Goldszmidt, 1997)	(Benferhat et al., 2008)
AODE	(Webb, Boughton & Wang, 2005)	(Koc & Carswell, 2015)(Baig, Shaheen & Abdelaal, 2011)
GBN	(Koller & Friedman, 2009)	(Bringas et al., 2008)

Table 3-1 presents NB and its variants that are discussed in Section 3.5 and the studies where they have been implemented in the intrusion detection domain. (Benferhat et al., 2008) applied the TAN algorithm on the KDD dataset. (Koc & Carswell, 2015) recently used AODE classifier in detecting DoS attacks too.

The aforementioned studies produced promising results of the application of Bayesian network classifiers, however they are evaluated using different methods. With respect to the literature discussed, there has been no research done to compare the Bayesian based classifiers performance in an intrusion detection problem. This research work presents a platform where they can be compared in detecting only DoS attacks in cloud computing environments.

3.8 Classification performance of Bayesian Networks Classifiers

3.8.1 What is classification?

Data mining is a powerful analytic process crafted to discover patterns and systematic relationships among variables from different perspectives with the purpose of extracting useful

information (Duda, Hart & Stork, 1997; Mitchel, 1997; Haghanikhameneh et al., 2012). Classification, also known as classification learning, is a major data mining technique used to explore and find the hidden knowledge in a dataset. A classification problem essentially involves categorising objects into groups according to their similar observed features (Melton et al., 1999). The problem aims to learn the relationship between the features variables and a target variable of interest. The problem of classification has been applied in a variety of data mining applications. Aggarwal (2014, p. 2) states the classification problem as follows: *"Given a set of training data points along with associated training labels, determine the class label for an unlabelled test instance"*.

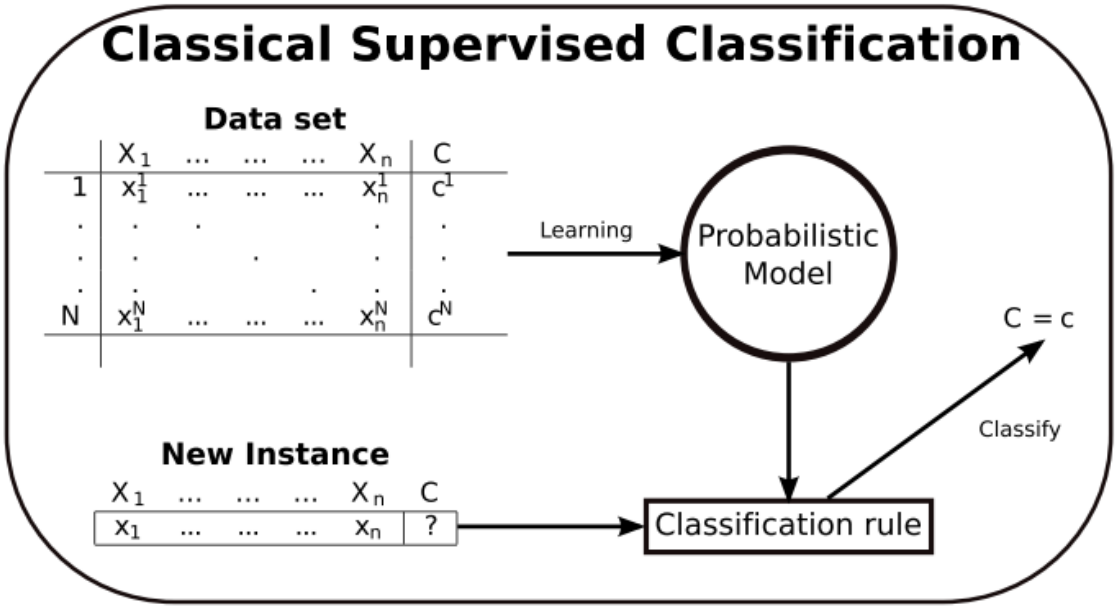


Figure 3-5 : Supervised Classification based on probabilistic classifiers(Rodriguez, 2013)

There are many variations in the definition of the classification problem discourse over varying settings. (Duda, Hart & Stork, 1997; Mitchel, 1997; Hastie, Tibshirani & Friedman, 2008) provide comprehensive overviews on the composition and application of data classification. Data classification is basically a two-step process as shown in Figure 3-5. In the first step, a classification model, mostly known as a classifier is constructed from a predetermined set of labeled examples. This is the training phase where a classification algorithm learns from a training set consisting of dataset instances and their associated class labels. An instance $\boldsymbol{X}$ is represented by an $\boldsymbol{n}$ attributes, $\boldsymbol{X} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n})$. Each instance is assumed to belong to a

predefined class and is determined by the additional dataset attribute known as the class label attribute. Because the class label for each training instance is provided, this renders this phase as supervised learning (Love, 2002; Sathya & Abraham, 2013).

The first phase of a classification task is the learning of a classifier, a mapping or function, $\boldsymbol{c} = \boldsymbol{f}(\boldsymbol{X})$, that is capable of assigning an associated class label $\boldsymbol{c}$ to a given instance $\boldsymbol{X}$. Classification tasks can either be binary (classifying an instance into one of the two classes) or multiclass (classifying an instance into one of the more than two classes) . Our research focuses on binary classification. A binary classifier assigns a class $\boldsymbol{c} \in \{\mathbf{0}, \mathbf{1}\}$ to an instance $\boldsymbol{X}$ based on its attributes $\boldsymbol{x} \in \boldsymbol{X}$.

In the second phase, the function is put to the test and used for classification. Classification performance can be defined as the ability to assign a class to a novel observation (Marcot, 2012). The strength of Bayesian networks classifiers largely depends on how well it categories the test instances after the completion of the training phase.

3.9 Classification performance – Bias and variance trade-off

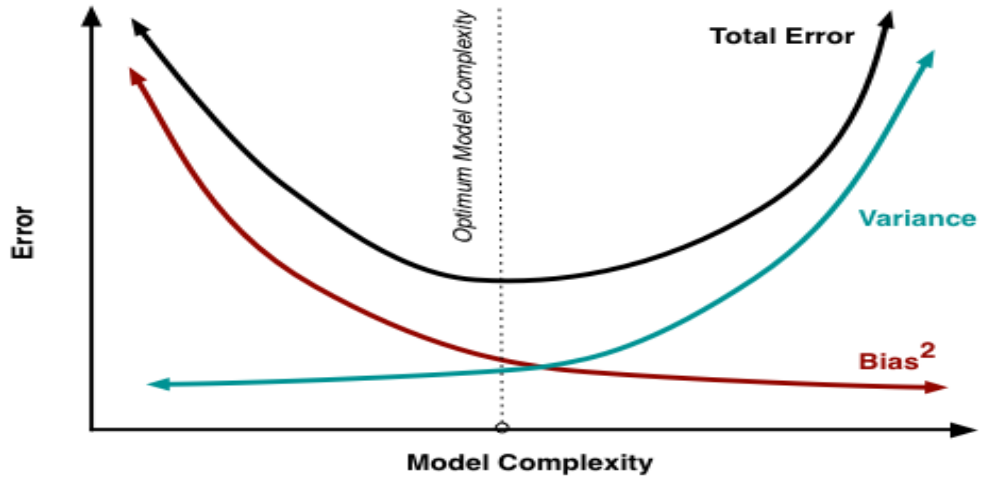


Figure 3-6 Bias and variance trade-off

The last two decades has seen a plethora of research exploring the factors that impact the classification performance of Bayesian network classifiers. The concept of bias and variance is inescapable with regards to the performance of classification algorithms. When a Bayesian model that is too complex given the training dataset is inducted, this leads to over-fitting

attributed to too much variance in the model. However, if a simple Bayesian classifier is learnt, it cannot extract a true structure given the data leading to under-fitting the data therefore there is too much bias in the model. To enhance the performance of Bayesian network classifiers, it is imperative to minimize either bias or variance. As shown in Figure 3-6, a bias/variance compromise has to be obtained for optimal classification performance of a learner (Brain & Webb, 1999).

It has been hypothesised and empirically investigated that the listed factors below are to a certain extent attributed to the over-fitting or under-fitting of Bayesian models (Brain & Webb, 1999; Castillo & Gama, 2005):

- Structural complexity of Bayesian network classifiers
- The training sample size
- The type of discretization approach
- The choice of score metric

The aforementioned factors, each has unique bearing of the classification performance of Bayesian network classifiers. It is however interesting to investigate how each factor impacts Bayesian models in classifying novel instances. In academia, ample studies have been conducted to date however they are not conclusive or binding. We review accordingly the literature in the following section.

3.9.1 Network structure: Model/Structure Complexity

There have been ample studies conducted in statistical modelling on the trade-off between parsimony and accuracy in prediction tasks (Marcot, 2012). A major concern is the determination of the appropriate classifier complexity so as to achieve optimal performance in a problem domain. In principle, many machine learning practitioners believe that an increase in model complexity should result in improve classification performance. Bayesian network classifiers can be arranged hierarchically with the order of structural complexity, as from NB to the most complex namely Bayesian multinet (Bielza & Larrañaga, 2014). The Bayesian classifiers investigated in the study can be organized as shown in Figure 3-7. The question stands: "*Does an increase in structural complexity learn a good classifier in a particular dataset?*". This might seem like obvious phenomena however empirically it could be otherwise.

Traversing the model complexity hierarchy, the independence supposition in simple NB is relaxed by addition of arc amongst variables. It also has an effect on the bias and variance of the model and ultimately the focus of this research on the classification performance of the model (Flores et al., 2011). A more elaborate motivation for this investigation, with respect to the no-free-lunch (NFL) theorems, the behavior of classifiers is not universal (Wolpert & Macready, 1997; Bielza & Larrañaga, 2014). Bielza & Larrañaga (2014, p. 35) states that it "*depends on the dataset".* Furthermore the authors point that it is significant to investigate the behavior of the whole hierarchy of Bayesian network classifiers. Numerous studies have been conducted on the comparison of the different classes of Bayesian networks to determine their classification accuracy. One such study was conducted (Bielza & Larrañaga, 2014). The authors did an empirical study on twelve different Bayesian network classifiers with varying model complexities. They trained their classifiers on Ljubljana breast cancer dataset with sample size of 286 instances. The results obtained are found in Table 3-2. Their results indicate that "*increasing model complexity does not necessarily imply a better model does not necessarily imply a better model"* (Bielza & Larrañaga, 2014, p. 32).

Table 3-2: Results from experiments done by (Bielza & Larranaga, 2014)

Classifier	**Classification Accuracy**
NB	71.64
TAN	77.57
BAN	74.78
Markov blanket-based Bayesian classifiers	75.16

Cheng & Greiner (1999) also did a study on the classification performance of Bayesian classifiers. They did an investigation on NB, GBN, TAN, CL multinet classifier, C4.5 and Selective Naïve Bayesian classifiers. The experiments were run on 25 datasets. One of the aims of this empirical study is "*improve the performance of the naïve Bayesian classifier by relaxing these independence assumptions"* (Cheng & Greiner, 1999, p. 2). Therefore this is basically an investigation on the progression in structural complexity of the Bayesian classifiers. From their results it is observed they are at variance with concept of universal optimal performance of algorithms. The simple NB classifier at particular dataset exhibited superior performance over its enhanced versions.

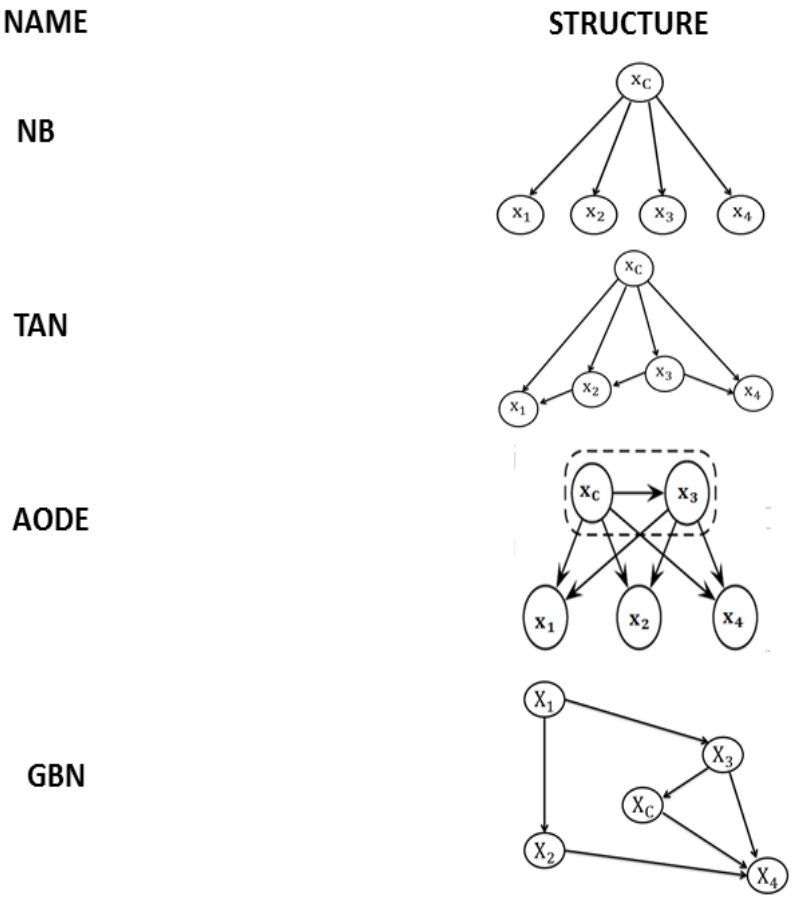


Figure 3-7: Summary of Bayesian network classifiers (Bielza & Larrañaga, 2014)

Madden (2009) conducted an empirical examination of the performance of some Bayesian classifiers namely NB, TAN and the GBN. They applied the aforementioned classifiers on 28 standard benchmark datasets. Their findings revealed that the GBN classifier significantly outperformed the NB with its performance closely equivalent to that of TAN. Table 3-3 provides a partial synopsis of their results.

Table 3-3: Results on classification performance of Bayesian network classifiers (Madden, 2009)

Dataset	**NB**	**TAN**	**GBN-K2**	**GBN-HC**
Adult	84.03 ± 0.53	86.15 ± 0.35	86.16 ± 0.33	86.02 ± 0.48
Car	85.15 ± 2.74	93.96 ± 1.90	89.61 ± 2.20	86.36 ± 3.15
Soyabean-large	91.83 ± 3.50	92.35 ± 3.08	89.22 ± 4.22	78.02 ± 6.45
Waveform-21	80.90 ± 1.64	81.96 ± 1.70	81.67 ± 1.56	79.73 ± 1.96
Australian	85.80 ± 4.03	85.06 ± 3.90	86.22 ± 3.83	85.93 ± 4.06
Lymphography	82.16 ± 10.61	81.07 ± 9.57	77.46 ± 9.47	75.06 ± 10.98

Table 3-3 depicts the progression in complexity from NB to GBN and their corresponding accuracies in the respective domains. Madden (2009) points out that GBN significantly outperforms NB on average. After a more rigorous analysis of their overall results reveals that the increase in model complexity does not necessarily guarantee improved classification performance. There are however instances of datasets such as Soyabean-large and Australian where we take note that the NB performance is good. Despite the dominance of the GBN over NB, there are domains in which the latter is the suitable choice of a classifier. Careful analysis of their results of the comparison between TAN and GBN reveals that there is no significant difference in term of classification accuracy. They propose an explanation of minor superiority of TAN in terms of the computational complexities. It is concluded that it more computational expensive to construct a GBN classifier as compared to a TAN.

Furthermore (Baesens et al., 2004) did a study towards the classification performance of Bayesian network classifiers in predict the future spending pattern of a newly acquired customer based on their first purchase. Their results are shown in Table 3-4.

Table 3-4: Results on classification performance of Bayesian classifiers (Baesens et al., 2004)

Classifier	**Training (%)**	**Testing (%)**
NB	71.0	72.5
TAN	74.9	74.0
GBN	75.3	75.0
GBN multinet	70.6	72.3

From Table 3-4, note that with progression in complexity from NB to GBN multinet there is some notable increase in the training accuracy however there is an irregular pattern in the test classification. GBN classifier had the highest classification performance on unseen data. This is at variance with finding of (Bielza & Larrañaga, 2014).

However, most of the experiments have been conducted on artificial data hence their findings might not be reliable when a real-life dataset is used. Section 3.5 provided a review of the Bayesian networks classifiers used in this study.

3.9.2 Training sample size

The number of instances in the training set at the induction phase has a notable impact on the predictive accuracy of Bayesian network classifiers. The training set plays a pivotal role in providing accurate probability estimates (Bielza & Larrañaga, 2014). Using bias-variance decomposition to explain how a finite training sample size n influences classification performance of a classifier:

1. Bias: a classifier inducted with n training instances performs poorly than a classifier trained with $n = \infty$ training instances.
2. Variance: a training sample size of n could build a classifier f different model performance. Also to note variance is inversely proportional to training sample size n.

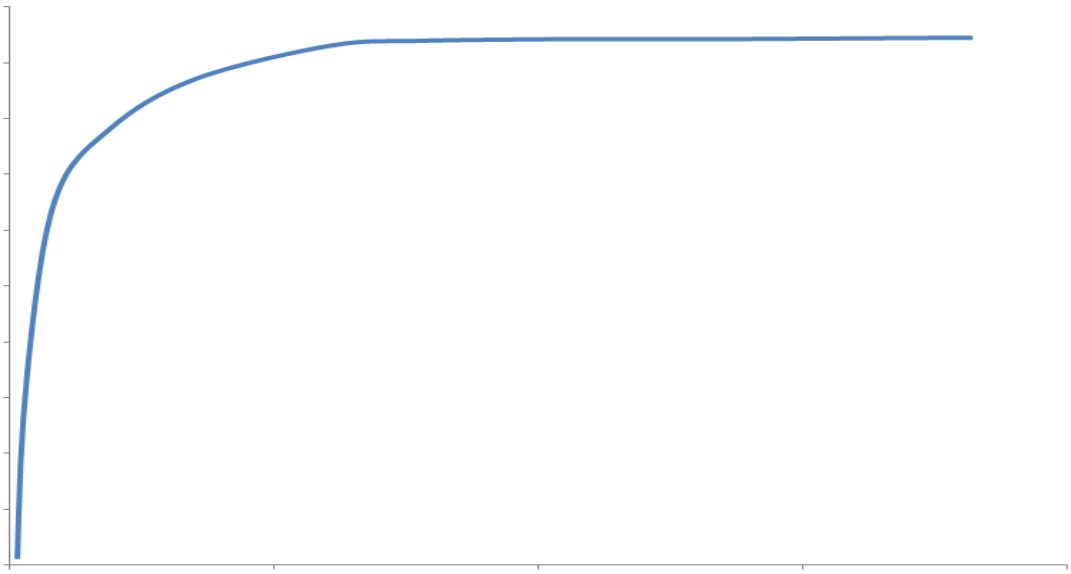


Figure 3-8: Generic learning curve (Figueroa & Zeng-treitler, 2012)

According to Beleites et al. (2013, p. 5), "*The learning curve describes the performance of a given classifier for a problem as function of the training sample size".* To emphatically explain how crucial the training sample size, it is imperative to make reference to an example of a learning curve depicted in Figure 3-8. Basically, the learning curve illustrates the relationship of computational cost and performance with increasing training sample data. There is a general agreement in the machine learning community that in the event of limited training sample size, the Bayesian network classifiers cannot populate the conditional probability tables with reliable

probability estimates (Bielza & Larrañaga, 2014). Also the larger the training sample size the joint probability distribution encoded to approximate the sampled domain (Hastie, Tibshirani & Friedman, 2009). In simpler terms, a large training sample contains more information. Also if the number of training instances is smaller as compared to the number of predictive variables this tends to lead to inaccurate classification rates. On the flip side, the number of training instances is linear on the time complexity of Bayesian network classifiers (Flores et al., 2011; Bielza & Larrañaga, 2014). It is thus crucial to have a training sample that is adequate for the induction process.

Numerous research studies have been done to explore the mechanisms towards sampling the optimum training sample data with respect to classifiers and ultimately their classification performance. This is highly ambiguous and there is bound to be significant disparities on the conclusions of respective publications.

Consequently this presents a gap during the learning process; the sampling of the training set and also how the size if the training sample set is impacted by other factors that are studied in the research i.e. discretization approach, score function etc. Furthermore traditional statistical classifiers such as Bayesian networks are said to more sensitive to the size of training sets compared to non-parametric classifiers such as ANNs and SVMs. Also general guidelines suggest that the number of training instances should be at least 6 times the number of predictive variables for optimum and accurate classification performance. Training sample sizes adversely affects the time, space complexities of the classifiers and ultimately their classification accuracies (Flores et al., 2011).

Bennett (2012) pointed out that according to the information theory; larger sample sizes contain more information than the smaller dataset of the same feature set. He explained this concept mathematically using the principal formula of the information theory using equation below:

$$\boldsymbol{H(X) = \sum_{i=1}^{n} p(x_i) I(x_i) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)} \qquad (\mathbf{3.14})$$

where $\boldsymbol{H}(\boldsymbol{X})$ represents the entropy value of $\boldsymbol{X}$; $\boldsymbol{p}()$ denotes the probability distribution; $\boldsymbol{I}()$ represents the self-information measure; $\boldsymbol{b}$ denotes the log base which is usually 2 and lastly $\boldsymbol{n}$ represents the sample size.

Bennett inferred from the Equation 3.14 that with the increase in $\boldsymbol{n}$, the training sample tends to approach the true sample size thus the probability distribution of the training sample size tends to approach the probability distribution of true sample. Furthermore, he pointed that increasing the sample size tends to obviate the impact of outliers and the perceived arbitrariness embedded in small datasets.

There is limited literature on the empirical studies pertaining to the correlation between the training sample size and the classification performance of Bayesian network classifiers. Scholars allude that there is a trade-off between the classifier accuracy and introduction of bias in relation to the size of the training set. Some researchers have pointed that good classifiers can be inducted from small training sets especially taking into consideration the cost of supervised learning data. On the other hand, some scholars alludes that a decrease in the number of training instances tend to produce inaccurate values particularly with regards to statistical models such as Bayesian based models. Therefore there is some form of discrepancy with regards to the opinions. The lack of adequate literature that is solely focused toward the impact of real sample size on the classification of Bayesian network classifiers motivated this object of our research.

Rossi (2013) in his study explored the effect of the training sample size on accuracy of NB and TAN classifiers. The classifiers were trained with sample sizes ranging from 10 to 435 of the Congressional Voting Records dataset from UCI. At small training size, NB outperformed TAN however as training sample size increases, TAN has a superior performance over NB. Rossi (2013, p. 5) attributed this to the fact that "*models learned with a small number of training examples have a very high variance but as the training data increased the variance decreases significantly"*. Bielza & Larrañaga (2014) points out that the independence supposition in NB works well with a small training sample size.

A typical study conducted by Sordo and Zeng (2005) who investigated the impact of sample size on classification accuracy of three classification algorithms NB, SVM and decision trees in classifying smoking data. NB was trained with sample size ranging from 50 to close to 8000

instances. As the training set size increases, the classification ability behaved asymptotically about 85%. They revealed that sample size is seemingly irrelevant with regards to classification accuracy. They discovered that the NB's accuracy stabilizes at a 4000 sample size threshold and thereafter remains invariant to some degree. The results depict that the most variation in accuracy occurs between 100 and 1000 samples.

Another study done by Rossi on the use of two Bayesian classifiers, NB and TAN in gene classification. Their training set had a maximum of 400 instances. Their results reveal that as the sample size increases, the classification accuracy increases from 70% to close to 95%. They concluded that by addition of the training instances that the classification accuracy significantly improved. Their work provided a comprehensive discussion that accounts for low accuracies at small sample sizes. Rossi pointed out that limited samples sizes are fraught with high variance and as the training set increases the variance decreases significantly. As the previous scholars, Rossi shows that the classification accuracy increases monotonically after a specific threshold training sample size. This pattern is typically of both the classifiers under investigation.

Domingos & Pazzani (1997) in their paper, "*On the Optimality of the Simple Bayesian Classifier under Zero-One Loss"* states rather striking findings pertaining to the relationship between the sample size and the predictive power of Bayesian classifiers. They point out that the classification accuracy of Bayesian classifiers does not necessarily increase with the increase in the number of training samples. They conducted empirical studies with artificial domain data with varying training sets. Their finding reveal that a sample size of 250 produced an accuracy of 84% while 500 instances of Diabetes data had an accuracy value of 75.5%. A Mushroom dataset of 800 instances has 94% accuracy however Wisconsin cancer of 500 sample size had an accuracy of 97.3%. Therefore classification accuracy is also domain-dependent. It is thus interesting to note the behavior of TCP/IP network dataset with regards to the sample size. They rather concluded with a contrary hypothesis that stipulates that Bayesian classifiers performs optimally when the sample size is small.

There is no shadow of doubt on how the training sample set plays a pivotal role in the classification performance of Bayesian network classifiers. The relationship between the sample size, the type of model and lastly classification performance must not be overlooked. It is fundamental to determine the amount of training size that will result in optimal performance of

Bayesian network classifiers. Being mindful of the cost of producing a labeled dataset for supervised learning purposes, it crucial to be able to determine that critical sample size that results in optimal performance. This might assist in evading trial and error that will introduce subjectivity concerning the appropriate sample size. Since the different Bayesian classifiers are built using different algorithms hence having varying structural, space and lastly time complexities, it would be interesting to note how they behave with regards to the sample size.

3.10 Induction phase: Score functions

The problem of building Bayesian networks remains a major research area. A recap of Section 3.6, the induction of Bayesian network is two-fold. The first step is for search for the network structure. There are numerous learning algorithms for constructing Bayesian networks from data. They come in two categories (Dally, Shen & Aitken, 2011): (1) conditional independence tests (2) score+search. Search+score techniques have received considerable attention in the machine learning community. This research is interested in algorithms that belong to the score+search paradigm.

3.10.1 Score+search algorithms

These algorithms approach the inductive learning process as a combinatorial optimization problem (Campos, 2006; Brenner & Sontag, 2013). Due to the "hardness" of this learning problem, heuristics methods are used to traverse the search space (Dally, Shen & Aitken, 2011). The search strategy involves seeking a structure from a pool of candidate models and the algorithm searches for the model that best fits the data based on the scoring function. The crucial task is to search for the highest-scoring network (Chickering, Heckerman & Meek, 2013). The structure learning problem is an optimization problem. The search space of potential network structures is a combinatorial space and contains a super-exponential number of network structures (Heckerman, Geiger & Chickering, 1994). Depending on how the score is defined, the learning problem should search for a structure that maximises or minimizes the score. One of the crucial decisions to be taken during the training of Bayesian networks is the type of scoring function. The score is a trade-off between how well the network structure best fits the data and the complexity nature of the network. Some score functions possess the decomposability property (Liu, Malone & Yuan, 2012). A scoring function is decomposable if the score value for

a network structure can be expressed as a sum of scores of the individual variables in the network; furthermore the score of the variable is computed based on the variable and its parents.

Thus,

$$\boldsymbol{Score(S|D) = \sum_{i=1}^{n} Score(X_i|\mathrm{Pa}_i, D)} \qquad (\mathbf{3.15})$$

Where S denotes the Bayesian network structure, D represent the training data.

The learning problem seeks to find S^* , where

$$\boldsymbol{S^* = \underset{S}{\mathbf{argmax}}\, Score(S|D)} \qquad (\mathbf{3.16})$$

As defined earlier, a Bayesian network structure encodes the joint probability distribution. Two structures are said to be class equivalent if they encode the same joint probability distribution (Liu, Malone & Yuan, 2012). A score function is said to be score equivalent if it assigns the same score to network structures that belong to same equivalent class (Yang & Chang, 2002; Liu, Malone & Yuan, 2012). Carvalho (2009) states that for optimality, it is imperative for score+search methods to possess the property of decomposability. In the same vein, for an efficient search in the context of equivalent classes of network structures, scoring functions must be score-equivalent too.

In the last decade, the induction of Bayesian networks from data has attracted significant attention in the field of artificial intelligence. To this end, as discussed in Section 3.6, algorithms have been developed to learn Bayesian networks from a database. The basic intuition of the algorithms such as K2 is to compute the joint probability of the Bayesian networks structure given the data and then thereafter create the most probable network structure that will maximise the joint probability. Thus, a scoring function plays the role of calculating the joint probability of the network structure given the data (Yang & Chang, 2002; Liu, Malone & Yuan, 2012). Unfortunately, the candidate structures are super-exponential in the number of variables of the model. The learning process is thus an intractable optimization problem especially if the number of variables is big e.g 5 variables will produce 29 281 structures while 10 variables produce 4.2 $\times 10^{18}$ network structures.

Most scoring metrics are created based on the assumption of network parameter distribution. Cooper (1992) coined two scoring metrics UPSM and DPSM based on the uniform distribution and general Dirichlet distribution respectively. (Bouckaert, 1994) derived the Minimum Description Length (MDL) to evaluate the quality of a network structure. The likelihood equivalence assumption which can also be referred to as Dirichlet distribution was proposed by (Heckerman, 1996) and produced the BDe score metric. Chang introduced the conditional uniform distribution and it derived the CUPSM score metric (Yang & Chang, 2002). Yang & Chang (2002) proposed that the network parameters are a subset of a valid prior. They believed that network parameters selected based on the different priors. They further alluded that different priors may perform differently under certain conditions. Liu, Malone & Yuan (2012) proposed that priors have different underpinnings hence they will output different optimal network structures.

Score metrics are introduced independently and the success of each is not an obvious universal venture. Yang & Chang (2002) suggested that their success varies with each application. Structure learning is thus engulfed by a mystery on the choice of scoring function to use in a particular application (Dally, Shen & Aitken, 2011; Liu, Malone & Yuan, 2012). Score-based approach to learning Bayesian networks has been proved to be NP-hard (Chi ckering, 1996). The basic idea behind a Bayesian scoring function is to evaluate the posterior probabilities of candidate networks S constructed from pre-knowledge of prior probability distribution given the training data $D, P(S|D)$. The concept behind the score+search algorithms for learning Bayesian networks can be simply expressed as follows as adapted from (Campos, 2006, p. 4):

Given a complete training dataset $D = \{u^1, \dots, u^N\}$ *of instances of* U_n*, find a DAG S* such that*

$$\boldsymbol{S^* = \underset{S \in \mathcal{S}_n}{\operatorname{argmax}}\, g(S : D)} \qquad (\mathbf{3.17})$$

g(S : D) represents the scoring function that evaluates the degree of fitness of any candidate Bayesian network model S to the data and S_n is the pool of all candidate models defined in U_n.

3.10.1 Score functions for Learning Bayesian networks

Literature provides several scoring functions for learning Bayesian networks. They are categorised into two (Yang & Chang, 2002; Liu, Malone & Yuan, 2012):

- ❖ Bayesian
- ❖ Information-theoretic

Bouckaert (2008) provides a comprehensive synopsis of the scoring metrics used in this study. Following is a brief description of the equations that describe the scores.

3.10.1.1 Akaike Information Criterion (AIC)

Bozdogan (1987) provides a comprehensive description of AIC. The AIC scoring function calculation is derived based on the asymptotic behavior of models having sufficiently large datasets. Given a pool of models for a given set of data, AIC evaluates the quality of each candidate model given other candidate models. AIC is an application of the information theory consequently it is used statistical modeling to estimates information loss in learning a model given its data. Hence during model selection, AIC handles the trade-off between goodness of fit and the complexity of a model. To further conceptualize the concept, we will use mathematical equation adapted from (Bouckaert, 2008). Let D represent a dataset and S to denote the network structure. Furthermore, let $r_i (1 \leq i \leq n)$ denote the cardinality of x_i. Let also q_i represent the cardinality of the parent set of x_i in S. Let N_{ij} $(1 \leq i \leq n, 1 \leq j \leq q_i)$ to represent the number of instances in D for which the parent node for x_i takes the jth value. Thus N_{ijk} $(1 \leq i \leq n, 1 \leq j \leq q_i, 1 \leq k \leq r_i)$ to represent the number of instances in D for which the parent of x_i takes jth value while x_i takes kth value. Lastly N denotes the number of instances in D.

Let $E(S, D)$ be the entropy value of a network structure and dataset and be represent as:

$$\boldsymbol{E(S, D) = -N \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \frac{N_{ijk}}{N} \log \frac{N_{ijk}}{N_{ij}}} \qquad (\mathbf{3.18})$$

The number of parameters K be represented by

$$\boldsymbol{K = \sum_{i=1}^{n} (r_i - 1) \cdot q_i} \qquad \mathbf{(3.19)}$$

Finally, the AIC score metric $AIC(S, D)$ of a network structure S for a dataset D is represented

$$\boldsymbol{AIC\ (S, D) = E(S, D) + K} \qquad \mathbf{(3.20)}$$

3.10.1.2 Minimum Description Length (MDL)

The MDL scoring prior for Bayesian networks learning was introduced by (Lam & Bacchus, 1994). MDL belongs to the information-theoretic family of scoring functions. Its principal idea for inducting the best model is based on minimizing the data used to encode the network structure and the training data given the network structure. The network can be encoded by storing both the parents of each modeled variable and the set of conditional probabilities associated with each variable. Given that there are n variables in the problem domain and a variable has k parents, a memory space of $\frac{p}{2}\log N$ bits is required to list the parent variables and p denotes the number of individual probability values for all variables.

The MDL principle dictates that with increasing model complexity, the longer the encodings required. Liu et al., (2012) proposes that the penalty term for MDL tends to be greater when compared to those produced by other scoring priors hence the best network output from the MDL prior tends to be denser than the best-fit structure of other scoring functions. In the stead of inducting an optimal structure, the MDL output a network structure that minimizes the score.

The minimum description length metric $Q_{MDL}\ (S, D)$ of a Bayesian network structure S for a training dataset D is defined as

$$\boldsymbol{Q_{MDL}(S,D) = E(S,D) + \frac{K}{2}\log N} \qquad \textbf{(3.21)}$$

3.10.1.3 BAYES

BAYES originally known as K2, is the first Bayesian scoring function introduced by (Cooper & Herskovits, 1992b). The Bayesian metric of a Bayesian network structure G for a training dataset D is

$$\boldsymbol{Q_{BAYES}(S,D) = P(S)\prod_{i=0}^{n}\prod_{j=1}^{q_i}\frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij}+N_{ij})}\prod_{k=1}^{r_i}\frac{\Gamma(N'_{ijk}+N_{ijk})}{\Gamma(N'_{ijk})}} \qquad \textbf{(3.22)}$$

where $P(G)$ represents the prior on the network structure and $\Gamma(.)$ the gamma-function whereas N'_{ij} and N'_{ijk} represents the choices of prior on counts restricted by $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$. The BAYES score satisfies the decomposability property.

3.10.1.4 Bayesian Dirichlet equivalence uniform (BDeu)

The Bayesian Dirichlet (BD) scoring function was coined by (Cooper & Herskovits, 1992a). The algorithm calculates the joint probability of a network structure given the dataset. Unfortunately, the metric has setbacks in that it requires user input to parameterize all the possible variable-parent relationships. Also, it is does not possess the score equivalence property. To mitigate the aforementioned limitations, Heckerman & Chickering (1993) proposed the equivalent sample size also known as hyperparameter α. The hyperparameter is the source of the required parameters and the prior distribution. The resulting score is referred to as the BDe and it is score equivalent. In the event that the network structures are of the class, the prior distribution over the network structures is assumed to be uniform. Coupling the BDe score metric and the uniformity assumption results in the BDeu.

When $N'_{ijk} = {}^{1}/_{r_i} \cdot q_i$ and consequently $N'_{ij} = {}^{1}/_{q_i}$ leads to the Bayesian Dirichlet equivalent (BDe) metric. The BDe uses Bayesian analysis to evaluate a network structure given the training dataset. Chickering, Heckerman & Meek (1997) developed the BDe scoring method. BDe is based on the BD metric. The approach is characterized by a multinomial distribution that describes the conditional distribution of each random variable in the structure.

With the Bayesian Dirichlet equivalent with uniform prior (BDeu) method, the Dirichlet distribution is modified by creating a fixed or uniform distribution of prior variable states. The score assumes likelihood equivalence which implies that the training dataset cannot be used expected to discriminate equivalent structures.

The described score functions are common and are provided in the WEKA suite. Several studies have been conducted that focus on evaluating score metrics for learning Bayesian network. However, Liu, Malone & Yuan (2012) cites that their conclusions are flawed due to the following reasons:-

1. They used local search algorithms such as Greedy Thick Thinning algorithm for selecting network structures which output suboptimal structures or arbitrarily produced network structures.
2. They used randomly generated datasets as test data hence synthesized data is different from a real–world dataset.

Van Allen & Greiner (2000) conducted a comparative study on three score metrics AIC, BIC and cross-validation to investigate the trade-off between structural complexity and goodness of fit to the training dataset. The researchers used randomly generated *gold standard* Bayesian network structures and also arbitrarily populated the conditional probability tables of the networks. Their results indicate that AIC as well as cross-validation are best at evading over-fitting during model selection. Their study also investigated the behaviour of the score metrics with changing training sample sizes. BIC was found to exhibit optimally performance at large dataset sizes and relatively poor for small sample sizes.

Yang & Chang (2002) performed an empirical comparison on five scoring functions UPSM, CUPSM, DPSM, BDe and BIC. The authors performed their investigation on the common ALARM benchmark network. K2 search algorithm was used in learning the Bayesian networks. The results produced noted that UPSM, CUPSM, DPSM and BIC learnt true networks. The study suffered a similar setback as the previous research; the experiment was not performed on a real-world dataset.

Carvalho (2009) investigated the impact of scoring functions on local search algorithms for Bayesian networks. Their study considered information theoretic-based scoring functions such LL, AIC, MDL, NML and MIT as well as K2, BD, BDe and BDeu. The authors evaluated the score metrics in a classification task by inducting a TAN model. Their findings based on the UCI datasets indicate that information-theoretic based classifiers produce optimally performing models than their Bayesian counterparts. Furthermore, we noted from their results that the performance of the scoring function was not universally ideal across the datasets. Malone & Matti (2015, p. 1) in their paper stated that, "*there is no guarantee that a network which optimizes a score for a training set will generalize well to new data".*

Malone & Matti (2002) conducted experiments on UCI and *sam* datasets. The authors wrote a publication titled, "*Impact of learning strategies on the quality of Bayesian networks*". Their study partially discussed the discourse of score functions. Their results suggested that simple network generalize well and they concluded that score functions with a high complexity penalty such as BIC will yield model with good classification abilities. Dally, Shen & Aitken (2011) also did a study on the issues affecting the learning of Bayesian networks. The ambiguity of the score functions in the induction process was explicitly stated. Another group of researchers,

Shaughnessy & Livingston (2005) did a study on search+score algorithms. They observed that choice of score function greatly impacts the "true classifier" more than the local search algorithm. They did a comparison on three score functions BDeu, "Bayesian metric" and MDL. The authors concluded that the type of scoring functions used involves a trade-off between sensitivity and precision of the resulting model. As a closing note, the researchers prescribed that the choice of scoring function is highly depended on how the network will be used. Riesen & Serpen (2009) embarked on a study to develop a Bayesian networks classifier that would be used in prediction of a victimization attribute in National Crime Victimization survey. Their results revealed that predictive ability of classifiers specified with different score functions were hurt by the type of score metric chosen. We provide part of their results in Table 3-5.

Table 3-5: Results on the prediction of the Victimization attribute

Search Algorithm	**Training (%)**	**Testing (%)**
Local Hill Climber (-P 3 –N –S BAYES)	99.28	99.22
Local Hill Climber (-P 3 –N –S ENTROPY)	99.61	98.77
Local K2 (-P 3 –N –S AIC)	98.98	98.86
Local K2 (-P 3 –N –S BAYES)	98.93	98.88
Local K2 (-P 3 –N –S ENTROPY)	99.15	98.17
Local K2 (-P 3 –N –S MDL)	97.13	97.08

The above reviewed papers' empirical evaluations do not suffice to draw conclusions from that could be deemed universal. Also all the studies do not discuss how the decomposability and equivalence properties help pertaining to classification tasks since they amount to a reduction in computational cost (Chickering, Heckerman & Meek, 2013).

The plethora of novel score functions in various field certainly implies that there exist some intrinsic relationship between the choice of score metric and the classification ability of new observations. Studies on choosing the appropriate scoring function for a particular dataset remains uncertain hence the fundamental question still remains: *for a given dataset, which scoring function will be appropriate for learning a good classifier?*. Most studies that have been done are to prove that different score functions identify different true networks. Liu, Malone & Yuan (2012) even pointed that local search algorithms using the same score function learn equivalent networks, the only notable difference will be on algorithm's runtime and memory

usage. Literature on how a score function impact the classification ability when their respective true networks classify new observations is limited. Acid et al. (2004) on their study on emergency medical service data asserts that score function influence generalisation power of Bayesian network classifiers and further identifying that non-Bayesian metrics result in robust classifiers as compared to Bayesian metrics. However Acid et al. (2004, p. 223) explicit state that, "*these assertions cannot be generalized without extensive experimentation using many different datasets*". This therefore dictates that the answer to the question requires an empirical study.

3.11 Discretization

In the field of data mining and machine learning, data pre-processing is a crucial phase that will guarantee the success of an algorithm (Pyle & Cerra, 1999; Maletic & Marcus, 2005). Data pre-processing entails processes; data transformation, data cleaning, data reduction. Discretization is one of the data reduction techniques and it has attracted a lot of attention as a pre-processing approach in data mining (Murphy, 2001; Liu et al., 2002). In machine learning, the Bayesian learning methods approach the classification task with the assumption that example attributes are discrete in nature. This unfortunately does not hold true in real life datasets that contain significant information that comes in continuous format. We intend to make use of the inherent information without stalling the learning the process. Bayesian networks classifiers can only work with nominal or discrete attributes. At its basic state, a discretization process converts quantitative attributes into qualitative format. Therefore numerical attributes are transformed into discrete or nominal attributes of finite number of partitions. Each partition is thereafter associated with a numerical discrete value.

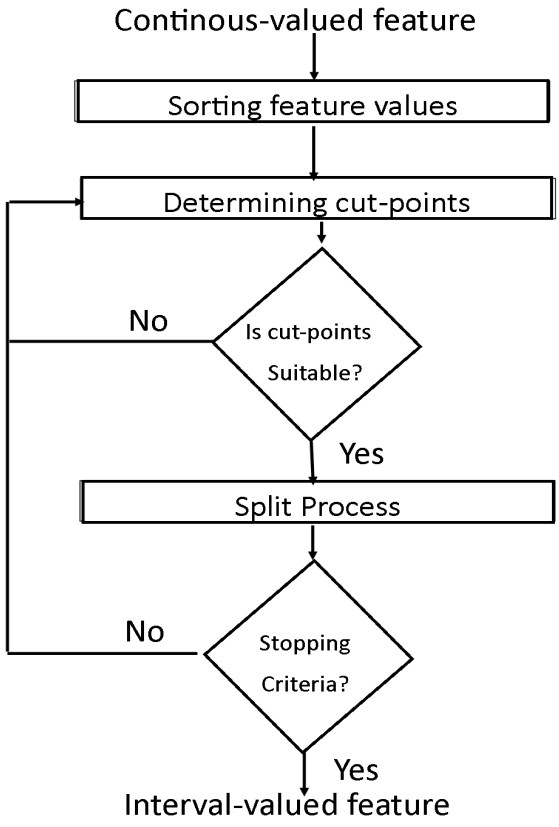


Figure 3-9 : Discretization process(Liu et al., 2002)

Discretization is defined as the process that transforms continuous variables into discrete variables (Dougherty, Kohavi & Sahami, 1995; Liu et al., 2002; Yang & Webb, 2002). Performing this aforementioned step requires the application of discretization methods to dataset inherently continuous in nature. Discretization of continuous variables has been investigated extensively in the following publications (Fayyad & Irani, 1993; Boullé, 2005; Divina & Marchiori, 2005). In supervised learning, classification in particular, discretization is defined by Mizianty, Kurgan & Ogiela (2008, p. 1) as follows: "*a discretization scheme D on a feature F converts the domain of values of F into d disjoint discrete subintervals, bound by a pair of values (boundary points)*

$$D: \{[d_1; d_2], \dots, (d_i; d_{i+1}], \dots, (d_{d-1}; d_d]\}$$

A typical discretization process follows a series of steps (see Figure 3-11). The process is usually conducted prior the induction process. Discretization is a necessary and important process due to several factors (Dougherty, Kohavi & Sahami, 1995; Liu et al., 2002; Maletic & Marcus, 2005):

1. Algorithmic requirements that are primarily designed to handle nominal attributes.
2. Results in data reduction which results in increased runtime.
3. Attributes of discrete nature are relatively human-readable, easy understand.

In the discretization context, a "cut-point" is a real value that exist within the range of continuous values and its partitions the range into two intervals. For instance, the continuous interval $[d_1; d_d]$ of the feature F is divided into $[d_1; d_2]$, $(d_2; d_3]$ etc, where a certain value d_i is a cut-point. The term cut point is also referred to as a split-point. The term arity refers to the number of partitions or intervals. Prior the discretization process, arity must be assigned a certain value k. a discretization process is allowed a maximum of k-1 cut-points. The ultimate goal of the discretization process is reducing the arity and there is a trade-off between arity and its impact on accuracy. A higher arity tends be unsuitable for the induction problem while a smaller arity is associated with a risk of information loss. Figure 3-10 illustrates how a continuous feature F_1 is transformed into a discrete F'_1 with values (V_1, V_2, V_3).

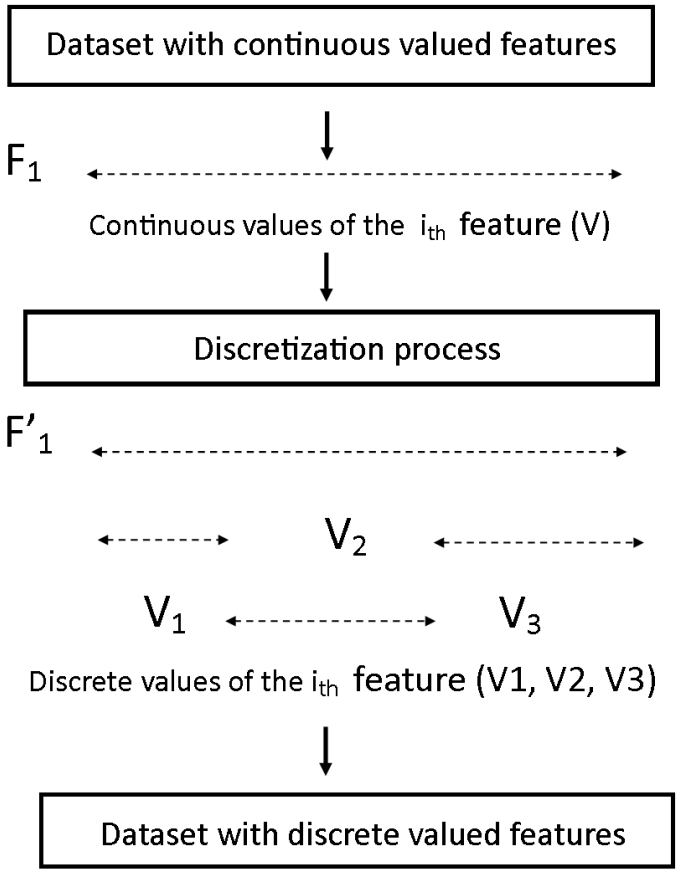


Figure 3-10: Transforming of continuous valued features into discrete (Hacibeyoglu, Arslan & Kahramanli, 2011)

However, despite discretization being of necessity, there is no standard approach to deal with the process. The nature of a discretization approach is impacted by the data set used, the model complexity (Sarkar, Srivastav & Shashanka, 2013). Despite outlining the criteria for optimality for discretization, the process is NP-complete (García et al., 2013; Sarkar, Srivastav & Shashanka, 2013) There are a vast number of discretization approaches that can be found in literature. García et al. (2013) alludes that the choice of a discretizer conditions the classification performance of a classifier.

Various heuristic algorithms have been proposed for discretization, schemes based on information entropy (Dougherty, Kohavi & Sahami, 1995), statistical (Kerber, 1992), likelihood (Wu, 1996) etc. The following section will further describe the criteria of discretizer.

3.11.1 Discretization methods classification

In accordance to the definition of discretization as a process of transforming or converting a large interval of continuous values into a set of finite values, each range of the original dataset is associated with a value in the discrete set; there are various types of methodologies with which this association is performed. Therefore there are at least three different axes in the classification of discretization approaches (Dougherty, Kohavi & Sahami, 1995). Discretization methods are classified as either:

- Supervised or Unsupervised- is the most discussed (Dougherty, Kohavi & Sahami, 1995; Liu et al., 2002; Hall et al., 2009)
    - Unsupervised methods also known as class-blind methods because they construct the discretization structure without involving the class attribute of an instance. This approach has possible limitations in that two data intervals of the discretization structure may overlap each other with regards to the class attribute value incorporated on both side of the interval division.
    - Supervised methods involve the class attribute in the discretization structure. With the knowledge of the class attribute, the interval division also referred to as the cut point will be more accurate rather than have values in the same range split in between.
- Global or local- the frequently discussed axes(Dougherty, Kohavi & Sahami, 1995)

    - Local approaches, such as the C.45 decision tree are inherent with a method of discretization within their internal structure. These approaches discretize subsets of the data that is within a particular section of the learning algorithm for instance a branch of a decision tree. Therefore the discretization algorithm is employed only on particular local instances rather than as a whole.
    - Global algorithms transform all the dataset instances in a single operation. Thus these approaches are on the front-end side of the learning phase.
- Static or Dynamic
    - Static methods are based on user input of the parameter that determines the number of cut points to be found in the dataset. The approach takes the data as input and determines the appropriate cut point at which to divide the data into k intervals. Each attribute is treated independently and binned according to its sub-intervals.
    - Dynamic approaches do not determine the number of intervals beforehand; rather the value of k will be derived from the dataset. During the discretization process the method uses a metric to evaluate the possible numbers of the cut point locations therefore the k assumes various values and subsequently chooses the value that optimizes the score metric for the final discretization process.
- Bottom-Up or Top-Down
    - Bottom-Up methods initiate the discretization process by first sorting the attribute data to be discretized and rendering each instance as a cut point. It traverses through the data set merging the instances and clusters of instances by getting rid of the cut points based on a certain metric. When the merging process elapses, substitution for values occurs.
    - Top-Down approach commences the discretization process with a single range for all the values of the continuous attribute data and employs an approach to determine additional cut points at which to partition the range. The binning process elapses when a particular condition is met.

Figure 3-11 illustrates a hierarchical decomposition of the taxonomy of discretization approaches. Various discretization approaches exist and many more are being developed. The

following subsection will discuss the discretization methods under study and review the state of the art criterions.

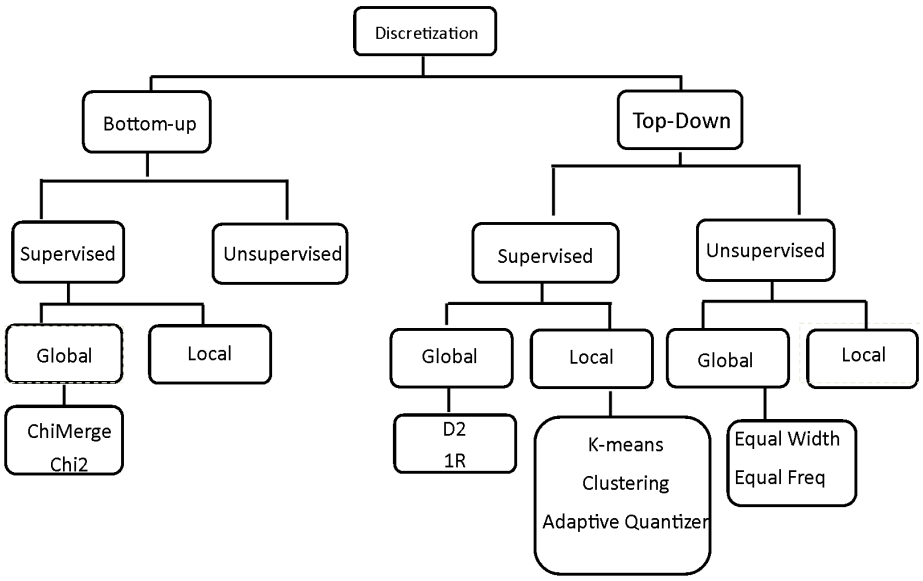


Figure 3-11: Decomposition of discretization methods

Empirical investigations conducted by several authors namely (Dougherty, Kohavi & Sahami, 1995), (Liu et al., 2002), (Clarke & Barton, 2000) found that in terms of classification accuracy, supervised discretization approach outperformed their unsupervised counterparts. The unsupervised algorithms tend to be arbitrary in terms of the division of variables. Hence the user determines the number of partitions. Their simplicity is attributed to the fact that a user is control of the resolution of attributes. They are preferably used in cases where "*there is not any information to determine relationships between variables"* (Hoyt, 2008, p. 6). A set-back of unsupervised methods is in their "class-blind" approach that tend to lead to over partitioning of the attribute values (Clarke & Barton, 2000; Hoyt, 2008).

3.11.2 Discretization criterions

Described in the following section are some of existent discretization approaches.

3.11.2.1 Equal Width Discretization (EW)

Equal Width Discretization also known as Equal Width Interval Discretization (Dougherty, Kohavi & Sahami, 1995) or Fixed k-Interval Discretization (Yang & Webb, 2001) is the simplest discretization method that is an unsupervised direct method. It divides the range of

observed attributes into k bins of equal size, where k is a parameter provided by the user (Dougherty, Kohavi & Sahami, 1995). If an attribute x is observed to have values bounded by x_{min} and x_{max}, then this method computes the bin width by:

$$\boldsymbol{\delta = \frac{x_{max} - x_{min}}{k}} \qquad (\mathbf{3.23})$$

Then the bin boundaries or thresholds are set at $x_{min} + i\delta$, where $i = 1, \ldots, k-1$.

This discretization method works independently of any multi-relational structure that could be inherent in the data. It is quite usual to set this value to 5 or 10 bins (Yang & Webb, 2001), although the optimum value for k depends on, among other factors, the dataset size. Its time complexity is $O(t)$, t being the number of data instances. For our empirical study, we will use a EW with k values set to both 5 and 10.

3.11.2.2 Equal Frequency Discretization (EF)

This discretization algorithm sorts values in an ascending order and divides the range into k bins so that each one contains approximately the same number of training instances (Dougherty, Kohavi & Sahami, 1995).

Hence, every bin contains t/k data instances with adjacent values. This discretization method provides a more balanced discretization by causing continuous values to be distributed into different bins. A group of data instances with identical values must be placed in the same bin therefore it is not always possible to generate k intervals with exactly the same number of values.

Time complexity for this algorithm is $O(t \log t)$ as it is necessary to perform an ordering of the data.

3.11.2.3 Fayyad and Irani's Entropy based Minimum description Length (MDL) Method

This technique is a supervised split method which is based on an entropy minimization heuristic to determine cut-points for discretization (Dougherty, Kohavi & Sahami, 1995). It uses a top-down approach whereby a numeric attribute is split to produce several discrete bins using a heuristic to decide on the number of intervals. The algorithm also uses the class information entropy of the numeric partitions to select a cut-off point that minimizes the entropy. The process is performed recursively by employing the Minimum Description Length (MDL) principle to

determine when to stop, which is when the MDL criterions or an optimal number of partitions has been satisfied.

The MDL principle subscribes to the notion *"that the best hypothesis is the one with minimal description length"* (Miltov et al., 2009, p. 33). The binning process tends to decrease the entropy value (Miltov et al., 2009). Recursive binning of an attribute with this criterion tends to result in the discretization of the attribute in question. The discretizer evaluates each candidate cut point which is the mid-point between successive attribute values. Furthermore during the evaluation of a cut-point, the attribute is discretized into two bins and thereafter the resultant class information is computed. In a case of binary discretization, a cut-point is chosen if its entropy is minimal when compared to other candidate cut points. The discretization is recursively applied and tends to select the optimal cut-point. The technique has a time complexity of $O(ct \log t)$.

3.11.2.4 Proportional k-Interval Discretization (PKID)

Proportional k-Interval Discretization is a discretization approach that was designed for the NB classifier (Yang & Webb, 2001). It aims to evaluate the trade-off between discretization bias and discretization variance (Boland, 2007). To explain the discretization bias and variance concepts we adapted the discussion from (Boland, 2007; Flores et al., 2011). The error component is categorised into bias, variance and an irreducible error. Bias is referred to as error due to the systematic error during the induction of the classification algorithm while variance is the error due to random variation in training data and random variation existing in the training dataset and from the random behaviour when inducting the algorithm. Classification variance is thus a metric to evaluate how a classification algorithm responds to variation in the training dataset. Discretization variance is thus the impact that a discretization approach has on the classification variance of a classification algorithm. To reduce the error, the number of intervals and the number of training instances must be considered. Intuitively, there is a conflict between reducing discretization bias and discretization variance. Discretization that results in fewer intervals will reduce variance. This means that the intervals will be large and therefore will accommodate more training instances. On the flip side, a discretization that produces more intervals will reduce bias. This is the source of the trade-off between bias and variance.

In light of the above discussion, Yang & Webb (2001) introduced PKID so as to provide a balance in terms of discretization bias and variance. It is similar to the EW, however, it does not have fixed number of intervals and the algorithm determines the number of intervals based on the number of training instances. Given that there are N training instances with known training attributes, PKID produces $p = \sqrt{N}$ intervals, each containing approximately $t = \sqrt{N}$ instances.

$$\boldsymbol{p \times t = N} \qquad (\mathbf{3.24})$$

$$\boldsymbol{p = t} \qquad (\mathbf{3.25})$$

where N represents the number of training instances, p represents the number of bins and t represents the number of training instances per partition.

3.11.2.5 Class-attribute interdependency maximization (CAIM)

Class-attribute interdependency maximization (CAIM) criterion takes a continuous attribute as input and transforms it into a finite number of intervals and thereafter associates each interval with a discrete value. This transformation tends to reduce the size of the data. The criterion is designed expressly to use class-attribute dependency information. In the stead of being dependent on user intervention, the CAIM approach automatically generates the number of intervals and computes the interval-width based on class-attribute interdependency values. The approach is associated with minimum number of discrete values as well as minimum loss of the class-attribute interdependency. In other words, this approach aims to generate intervals that maximize the class-attribute interdependency. Therefore CAIM makes use of the class-attribute interdependency information as a criterion for optimum discretization. For a detailed review of the criterion and the associated mathematics, the reader may refer to (Kurgan & Cios, 2001; Mizianty & Kurgan, 2010).

CAIM criterion is the underlying principle for the CAIM discretization algorithm. This discretization algorithm traverses the search space with the goal of finding the highest values of the CAIM criterion. CAIM algorithm makes use of the greedy approach since the search problem is combinatorial.

3.11.2.6 ur-Class-attribute interdependency maximation (ur-CAIM)

ur-CAIM is an extension of the CAIM algorithm. The ur-CAIM criterion (Cano et al., 2014), is a resultant of three class-attribute interdependence criterions that are designed to work together in a robust manner. For a comprehensive review and the associated mathematics of the ur-CAIM algorithm, the reader may refer to (Flores et al., 2011; Cano et al., 2014).

The above reviewed discretization approaches can be used for any classification task that involves Bayesian networks. The question however remains: *"Which discretization algorithm will be best for a particular dataset*?" The search for the best discretization for a classification task is hard. The choice of discretization approach is of a discretization algorithm is crucial as it tends to impact the efficacy and effectiveness of a Bayesian classifier (Flores et al., 2011). Discretization approaches also improves the runtime of Bayesian algorithms. However most importantly reflecting on the focus on this work to check how the choice of discretization approach impacts the classification ability of Bayesian networks classifiers. The type of discretization method whether supervised or unsupervised, the number of bins are the pertinent features for a discretization algorithm (Dougherty, Kohavi & Sahami, 1995; Flores et al., 2011). Each of these attributes impact the classification performance of a Bayesian network, we study each of the attributes. García et al. (2013) suggests that performing empirical experiments using a set of classifier models and discretization approaches help to identify the best performing approach. Furthermore, we examine how the training sample size impacts the discretization process and consequently the classification performance of Bayesian network classifiers.

3.11.3 The choice of discretization approach

As previously mentioned, the discretization process generally involves some form of data reduction hence leads to information loss. Therefore it is the ultimate goal to have a discretization approach that minimizes information loss. The selection of an optimally performing discretization method is NP-complete (García et al., 2013). There is a wealth of literature that discusses the use of discretization schemes, however these methods have not been compared on how they condition the classification in terms of accuracy.

Kaya et al (2011) conducted experiments on the diagnosis of Parkinson's disease. They did a comparison of non-discretized and discretized data on classification performance. They concluded that discretization does impact the classification accuracy of machine learning

techniques. Lee (2007) did a study on the relationship between the choice of discretization approach and classification accuracy. He notes that a discretization approach incurs some form of information loss that tends to lead to classification error. Furthermore, he alludes that poorly discretized attributes are likely to lead to inaccurate classifications. Freitas (2002, p. 50) states that, *"In practice, discretization may either increase or decrease classification accuracy, depending on the original data and on the data mining applied to the discretized data".* They also mentioned that discretization is a type of abstraction process hence relevant information tends to be lost during the process.

Flores et al (2011) did a study on the extent to which the type of discretization algorithm affects classification ability NB and its variants on 26 datasets. They concluded that a discretization approach produces different results for each dataset. They further stated that the discretization methods do not really affect the classifier ranking when compared. Tillander & Pavlenko (2009) also embarked on a similar study to investigate the relationship between the discretization algorithm and classification performance of high dimensional classifiers. Their study pointed that the discretization procedure affects the dependence structure of classifiers. Results revealed that supervised discretization method incur the lowest misclassification errors since they alter the dependence structure the most. Dougherty, Kohavi & Sahami (1995) also did a study titled, *"Supervised and Unsupervised discretization of continuous features,"* reveal that on average discretization methods lead to an increase in accuracy. However the effects of discretization approach is not universally positive because the performance might be improved in certain datasets and significantly degrades in other datasets. The authors note that it is significant to know whether the discretization algorithm is global or local since they have varying effect on the accuracy of classifiers. They also concluded that supervised learning algorithms are better than their unsupervised counterparts.

There is some form of uncertainty around the appropriate choice of discretization approach. García et al (2013) advised that the choice of discretization method is highly dependent on the domain under study and the data mining method employed. They allude that an empirical study is crucial to determine the suitable candidate.

3.12 Conclusion

This chapter gave an extensive review of literature and the first principles of Bayesian networks. The basics of Bayesian networks: the probability theory, their network structure, the conditional probabilities, the inference process were discussed in detail. This chapter presented a detailed overview of Bayesian network classifiers namely Naïve Bayes (NB), Tree-augmented naïve Bayes (TAN), Averaged one-dependence estimators (AODE) and General Bayesian network (GBN). The equations defining the classifiers were presented. Finally the chapter dedicated a section to discuss in the detail the application of Bayesian network classifiers implemented in the research in the intrusion detection domain. The discussions revealed the different approaches used in the training of the classifiers in detecting intrusions. However they did not provide a comparison of the classifiers in a similar set-up. The concepts about the training set size, discretization approach and score functions were not reviewed in these publications. The following chapter will deal with the concepts mentioned and how they are involved with Bayesian networks models. The chapter has introduced and discussed important literature that concerns the classification performance of Bayesian network classifiers. The review might not be entirely extensive and exhaustive however it acknowledged relevant scholarly work that pertains to the optimal performance of Bayesian network classifiers. The ultimate purpose of this chapter was to introduce the elementary knowledge available by referencing the relevant research works.

Chapter 4: Methodology and Design

This chapter gives the methodology framework followed in the design of the Bayesian networks classifier-based intrusion detection system. Presented, are the software tools used in capturing the datasets and discussion of the associated issues in the creation of the dataset. Section 4.1 describes the conceptual framework adapted for our study. Section 4.2 provides a detailed account of the training dataset, its composition as well as the source. It also describes the steps taken towards the live packet capture on the cloud platform. It outlines the data-pre-processing procedure done prior to the empirical study. Lastly the chapter presents metric for performance evaluation of the experiments done, followed by a conclusion and summary of the chapter in Section 4.3.

4.1 Methodology

Kothari (1990) in his book states that there are basically two approaches to research namely quantitative and qualitative. He further points out that the former can be categorised into inferential, experimental and simulation methods to research. Inferential approach entails forming a database from which to deduce the characteristics or relationships of the sample population. Usually the characteristics are determined through questionnaires or observations. Simulation approach is particularly used for research that involves complex phenomena and hence cannot be setup in a laboratory environment. Experimental research involves conducting empirical studies with the aim of obtaining results from a real-world test-bed. The experiments are largely for testing the veracity of formulated research hypotheses. This research method is the most used in computer science for performance and behaviour analysis. Experimental approaches are an important methodology for environments such as a network setup with multiple users accessing services. In this study we used the experimental research method. Figure 4-1 illustrates the experimental design process.

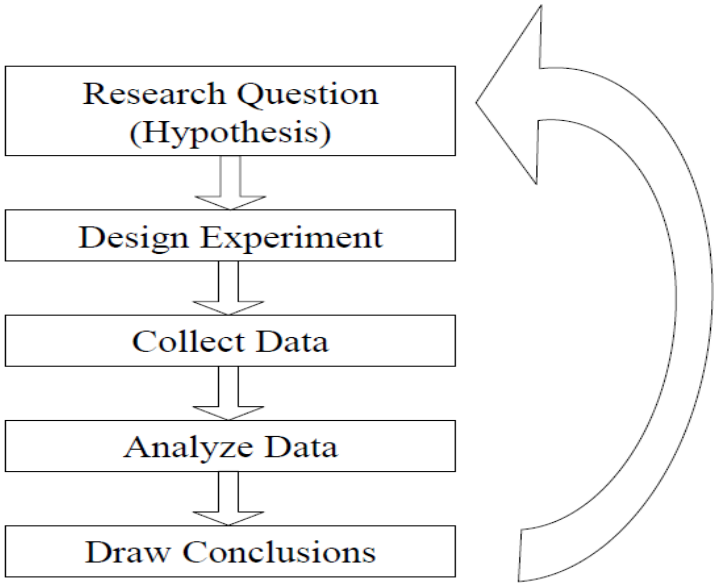


Figure 4-1: Research design

The design process stipulates that the researcher must first formulate the hypothesis. Research questions outlined are based on the motive of the research study. The object of the research is to test the hypothesis or answering the research questions posed. Subsequently, the design other experiments commences. The materials and technologies are described and acquired. The test-bed is setup and experiments are conducted. Collected data will still be in its raw form. Therefore, it was subjected to some analysis to produce information. Conclusions were drawn from the information extracted. Lastly, there is an assessment on whether the conclusions made answer the research question outlined initially. For this research, we chose the experimental approach because the core of our work involved the manipulation of variables in the model and observe what effect they will produce.

4.2 Research Design

The research methodology adopted in this work has been discussed in section 4.1. This sub-section provides experimental research design, which involves the details of the materials and procedures of our experiments.

4.2.1 Conceptual approach to BNC-NIDS

This study was conducted under the framework that is depicted conceptually in Figure 4-2. We made use of the NSL-KDD dataset for the intrusion detection for constructing the Bayesian classification models through supervised induction and evaluating the classifiers on test set. The

framework incorporates Bayesian networks classifiers under study as illustrated in the classification model module in Figure 4-2. The classifiers are implemented for a classification task for detecting DoS attacks as revealed in the Classification of network event module.

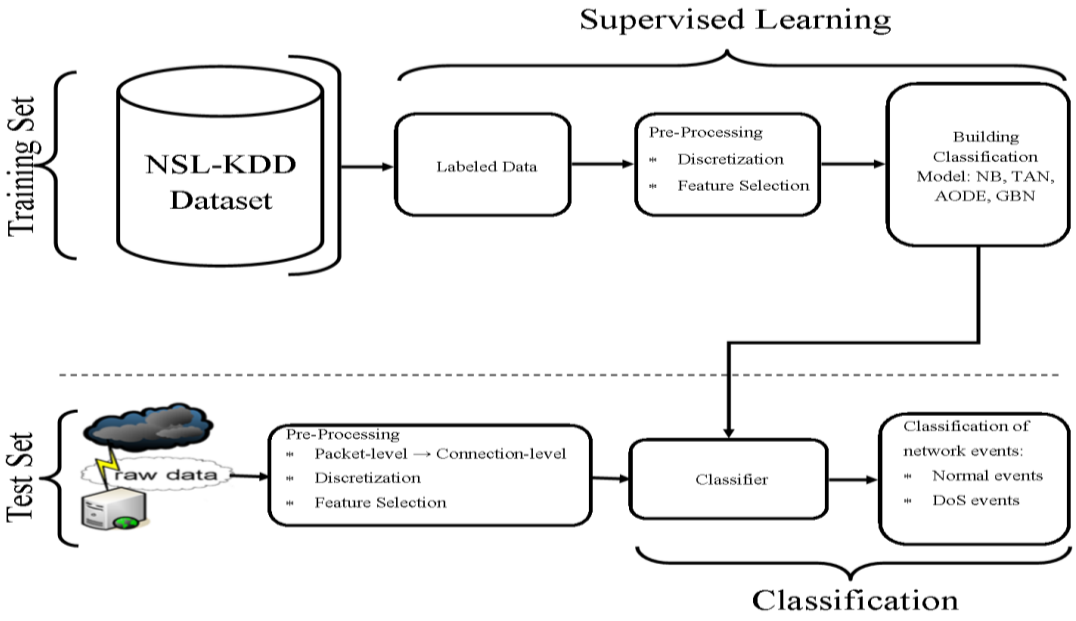


Figure 4-2: Conceptual Framework

Our empirical research design follows the steps below:

1. Training set and test set.
2. Perform data pre-processing.
3. Model induction from the NSL-KDD dataset by applying cross-validation.
4. Evaluation of the induced classifiers on the test set.

The first step entails the NSL-KDD dataset as the training set. The test set consists of the raw dataset captured on the cloud environment as illustrated on Figure 4-2. In-depth reviews of the NSL-KDD dataset and the collection of the test set on the cloud are provided in Section 4.2.3.

We conducted data pre-processing on both the training and test set in step 2. Firstly, the training set and the test set must be in the same format. The test set was in its raw format, packet level, while the training set was in the connection-level format. Therefore the test set was converted to convention-level using Wireshark. As discussed in Section 3.4 Bayesian networks only model discrete features. Our training and test sets consist of continuous features which therefore must

be discretized. With respect to the conceptual framework, in step 3 and 4 is where our experiments were conducted. The results gathered in the experiments will be presented in Chapter 5.

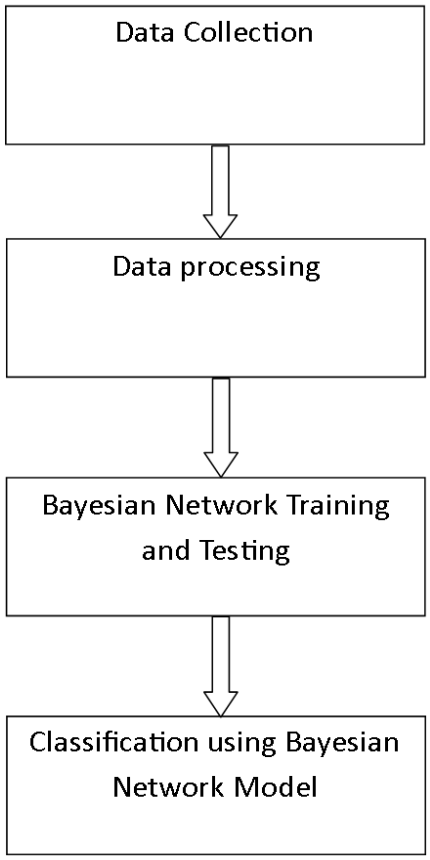


Figure 4-3: System flowchart used in the study

Figure 4-3 illustrates the systematic sequence followed in this study. Our empirical experiments started with data collection which plays a pivotal role in the evaluation of the constructed classifiers. Data-preprocessing, the most critical stage follows and it is imperative to ensure an accurate evaluation dataset. As reviewed in Section 3.4, due to the algorithmic requirements of the Bayesian networks under investigation, this study used the discretization process responsible for transforming the data attributes of continuous nature to discrete format. Lastly the Bayesian network classifier structures were constructed on the fourth step. Thereafter the experiments involve both the training and testing process of the built models.

4.2.2 NSL-KDD Intrusion Detection Dataset

In the year 1998 and 1999, Massachusetts Institute of Technology (MIT) Lincoln Labs in collaboration with Defence Advanced Research Projects Agency (DARPA) prepared the 1998 DARPA dataset and 1999 the DARPA dataset for use in Intrusion Detection field (Lippmann & Haines, 2000; Lippmann, Fried, et al., 2000; Lippmann, Haines et al., 2000). KDD-99 dataset is a sample of the DARPA dataset developed by Wenke Lee and Sal Stofo (KDD, 99). The KDD'99 training dataset that consist of TCPDUMP data simulated over a span of seven weeks of network traffic processed to connection–level format. A connection comprises of a sequence of TCP packets moving from a source host to a target host using a particular protocol. Each connection is labeled as either normal or as attack. Each record consists of 41 features, which are detailed in Appendix N and a target class (Lee, Stolfo & Mok, 1999). Features are grouped into categories:

- Basic features are the first six features that describe each individual TCP connection.
- Content Features: Domain knowledge is used to assess the payload of the original TCP packets. This included features such as the number of failed login attempts.
- Time-based Traffic Features: these features are designed to capture properties over a 2 second temporal window. One example of such a feature would be the number of connections to the same host over the 2 second interval.
- Host-based Traffic Features: Utilize a historical window estimated over the number of connections, for instance 100, instead of time. Host based features are therefore designed to assess attacks, which span intervals longer than 2 seconds.

The training dataset comprises of a total of 22 attacks categorized as follows:

- Denial of service (DoS): the attacker seeks to prevent legitimate end-users from accessing a service e.g. Neptune, Smurf, Pod and Teardrop.
- Remote-to-Local (R2L): unauthorized access of a local host from a remote machine e.g. Guess-password, Ftp-write, Imap and Phf.
- User-to_Root (U2R): unauthorized access to root privileges e.g. Buffet=overflow, Load-module, Perl and Spy.
- Probing e.g. Port-sweep, IP-sweep, Nmap and Satan.

Appendix N gives a description of KDD'99 Dataset feature and their respective data types (Kayacik et al., 2005). The dataset has been the widely used in IDS research studies (Sabhnani & Serpen, 2003). There exist inherent issues with KDD'99 Data and Tavallaee et al. (2009) provides an in-depth analysis of the inherent problems in that dataset. To address the issues, a new data set, NSL-KDD, was proposed. The dataset comprises of selected records from the KDD data set. However there have been many critiques and arguments against the use of this dataset (Mahoney & Chan, 2003; Brugger & Chow, 2007). The most highlighted issue is that there exists no validation to ensure that the NSL-KDD dataset is a real and authentic reflection of existing real network traffic. Unfortunately, due to the lack of contemporary and free public datasets, the NSL-KDD dataset was used as our common reference point. The data was downloaded and saved as an ARFF format which is compatible with WEKA software.

4.2.3 Data collection from the Cloud Computing Environment

The dataset used as the test set in this research study was prepared in a normal lab environment, under a proxy network at the University of Fort Hare. In order to acquire raw tcpdump network data for a Cloud Computing environment, a cloud deployment test-bed was set-up using open-source ownCloud software (OwnCloud, 2015). ownCloud is a file storage, sync and sharing platform that works in a similar way as dropbox. The test-bed comprised of three workstations. All workstations were running a Linux environment operating on Ubuntu 12.04. One workstation was used for the attacking role and other node was used by the legitimate cloud user. The remaining workstation served as a server in which we deployed our minimal cloud environment with the following specifications: Pentium(R) Dual-Core CPU T4400 @ 2.20GHz CPU, 80GB disk space and 3GB memory. Figure 4-4 depicts the general overview of the test-bed.

In our test-bed setup, data transmission between the workstations was attained through a LAN network with 100Mbps bandwidth. Data transmission was through a wired network. The rationale for our choice of a wired network over a wireless network in this research study was due to the fact that a wired network is stable and fast, hence the connectivity was more reliable (Isnin, 2011).

The first dataset collection experiment involved the consumption of cloud services. The process required that the user validated by entering their logging credentials. At this point, the client user would have been registered already into the cloud system.

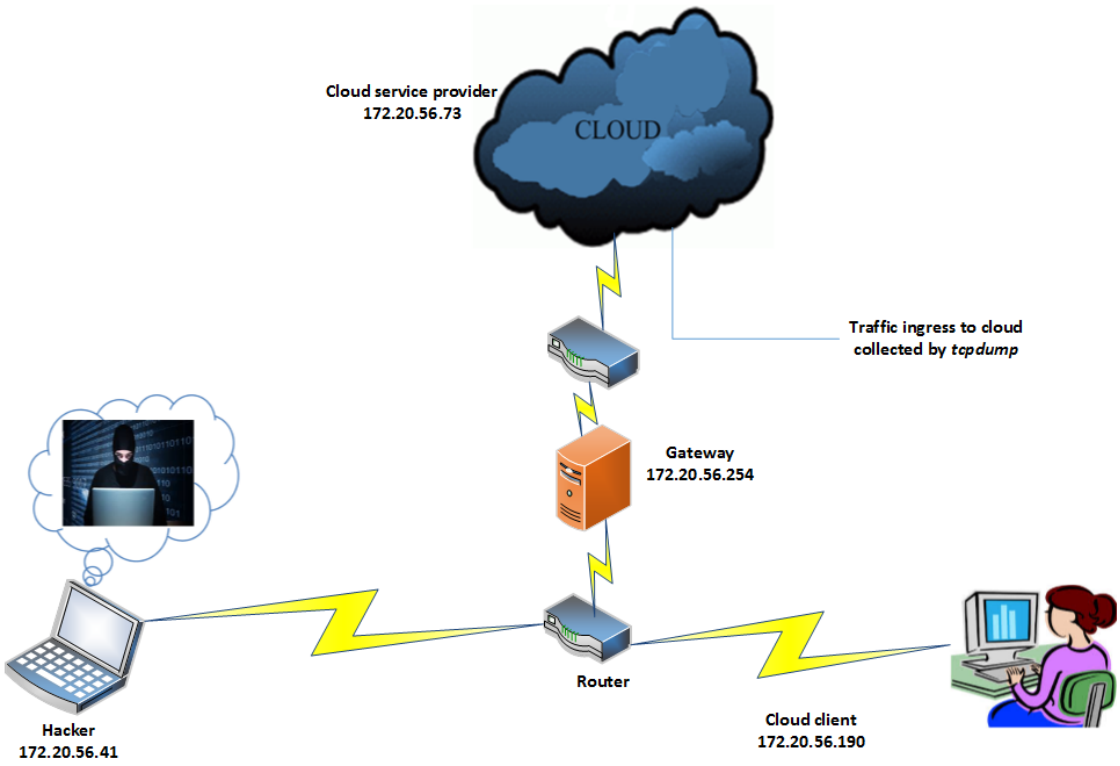


Figure 4-4: Overview of the design set-up

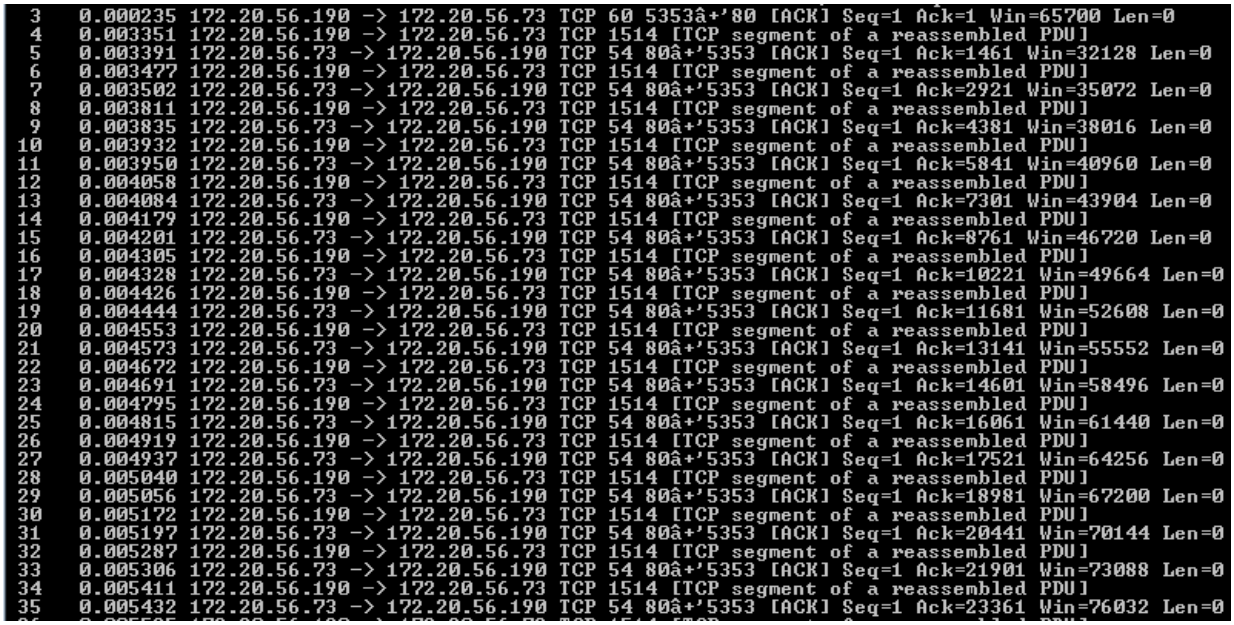
When the client user performs the file uploading, file deletion processes, the tcpdump software utility was invoked to sniff the network traffic data between the two communicating nodes (client and server machines). *Tcpdump* is an open source software package that is available at no cost. The advantage of using *tcpdump* in this empirical study was due to the fact that it inherently allows command expressions that permit the user to filter out various types of traffic and focus on the traffic required. Figure 4-5 illustrates the expression invoked for our experiment.

```
root@lindani-Aspire-E1-572:/home/lindani# tcpdump -w /home/corpus.pcap -n -vvv -i eth0 tcp && 172.20.56.73
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
^C203 packets captured
203 packets received by filter
```

Figure 4-5: Expression invoked for packet capture

The captured traffic data was saved as a *tcpdump* pcap file. The *tcpdump* pcap file contained a connection sequence of TCP packets starting and ending at some well-defined times, between

which data flows to and from a source IP address to a target IP address under a specified protocol. A sample of the raw *tcpdump* is shown in Figure 4-6.



```
  3   0.000235 172.20.56.190 -> 172.20.56.73 TCP 60 5353â+’80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
  4   0.003351 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
  5   0.003391 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=1461 Win=32128 Len=0
  6   0.003477 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
  7   0.003502 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=2921 Win=35072 Len=0
  8   0.003811 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
  9   0.003835 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=4381 Win=38016 Len=0
 10   0.003932 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 11   0.003950 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=5841 Win=40960 Len=0
 12   0.004058 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 13   0.004084 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=7301 Win=43904 Len=0
 14   0.004179 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 15   0.004201 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=8761 Win=46720 Len=0
 16   0.004305 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 17   0.004328 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=10221 Win=49664 Len=0
 18   0.004426 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 19   0.004444 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=11681 Win=52608 Len=0
 20   0.004553 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 21   0.004573 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=13141 Win=55552 Len=0
 22   0.004672 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 23   0.004691 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=14601 Win=58496 Len=0
 24   0.004795 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 25   0.004815 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=16061 Win=61440 Len=0
 26   0.004919 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 27   0.004937 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=17521 Win=64256 Len=0
 28   0.005040 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 29   0.005056 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=18981 Win=67200 Len=0
 30   0.005172 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 31   0.005197 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=20441 Win=70144 Len=0
 32   0.005287 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 33   0.005306 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=21901 Win=73088 Len=0
 34   0.005411 172.20.56.190 -> 172.20.56.73 TCP 1514 [TCP segment of a reassembled PDU]
 35   0.005432 172.20.56.73 -> 172.20.56.190 TCP 54 80â+’5353 [ACK] Seq=1 Ack=23361 Win=76032 Len=0
```

Figure 4-6: Sample of normal traffic activity

The second evaluation dataset experiment involved adding a third node for simulating DoS traffic to attack the cloud system. The DoS traffic is a type of attack whereby an attacker floods the cloud network with useless packets (Yu, 2013). DoS attacks investigated in numerous research works involving cloud networks (Shea, 2012; Alina & Popescu, 2013; Lonea, Popescu & Tianfield, 2013). During the simulation of the DoS attack, the motive of the attacker is to handicap the cloud services so that they are not available to the cloud clients. The attacker makes use of *hping3* attack tool developed by Salvatore Sanfilippo famously known as Antirez from Italy to perform the DoS attacks (Sanfilippo, 1998). *Hping3* is a packet generator and analyser for the TCP/IP protocol but for the purpose of this research it was only used for assembling custom packets. It can be easily installed on any Linux distribution using the synaptic manager. *Hping3* was used to initiate Layer 4 attack. A Layer 4 attack is realised by exploiting the three-way handshake in the client-server model. Figure 4-7 shows the command used to initiate a DoS attack in our experiments.

```
root@lindani-Aspire-E1-572:/home/lindani# hping3 --flood -p 80 -c 10 172.20.56.73
HPING 172.20.56.73 (eth0 172.20.56.73): NO FLAGS are set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 172.20.56.73 hping statistic ---
5917173 packets transmitted, 0 packets received, 100% packet loss
```

Figure 4-7: Hping3 command

```
  1   0.000000 172.20.56.190 -> 172.20.56.73 TCP 66 8508â+'80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
  2   0.000043 172.20.56.73 -> 172.20.56.190 TCP 66 80â+'8508 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
  3   0.001077 172.20.56.190 -> 172.20.56.73 TCP 60 8508â+'80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
  4   0.003318 172.20.56.190 -> 172.20.56.73 HTTP 557 GET /owncloud/index.php/apps/files/ajax/getstoragestats.php HTTP/1.1
  5   0.003357 172.20.56.73 -> 172.20.56.190 TCP 54 80â+'8508 [ACK] Seq=1 Ack=504 Win=30336 Len=0
  6   0.186155 172.20.56.73 -> 172.20.56.190 HTTP 579 HTTP/1.1 200 OK  (application/json)
  7   0.384446 172.20.56.190 -> 172.20.56.73 TCP 60 8508â+'80 [ACK] Seq=504 Ack=526 Win=65172 Len=0
  8   5.190209 172.20.56.73 -> 172.20.56.190 TCP 54 80â+'8508 [FIN, ACK] Seq=526 Ack=504 Win=30336 Len=0
  9   5.190668 172.20.56.190 -> 172.20.56.73 TCP 60 8508â+'80 [ACK] Seq=504 Ack=527 Win=65172 Len=0
 10   5.190953 172.20.56.190 -> 172.20.56.73 TCP 60 8508â+'80 [RST, ACK] Seq=504 Ack=527 Win=0 Len=0
 11 188.561131 172.20.56.41 -> 172.20.56.73 TCP 60 1936â+'80 [<None>] Seq=1 Win=512 Len=0
 12 188.561202 172.20.56.41 -> 172.20.56.73 TCP 60 1937â+'80 [<None>] Seq=1 Win=512 Len=0
 13 188.561267 172.20.56.41 -> 172.20.56.73 TCP 60 1938â+'80 [<None>] Seq=1 Win=512 Len=0
 14 188.561334 172.20.56.41 -> 172.20.56.73 TCP 60 1939â+'80 [<None>] Seq=1 Win=512 Len=0
 15 188.561401 172.20.56.41 -> 172.20.56.73 TCP 60 1940â+'80 [<None>] Seq=1 Win=512 Len=0
 16 188.561468 172.20.56.41 -> 172.20.56.73 TCP 60 1941â+'80 [<None>] Seq=1 Win=512 Len=0
 17 188.561535 172.20.56.41 -> 172.20.56.73 TCP 60 1942â+'80 [<None>] Seq=1 Win=512 Len=0
 18 188.561602 172.20.56.41 -> 172.20.56.73 TCP 60 1943â+'80 [<None>] Seq=1 Win=512 Len=0
 19 188.561670 172.20.56.41 -> 172.20.56.73 TCP 60 1944â+'80 [<None>] Seq=1 Win=512 Len=0
 20 188.561737 172.20.56.41 -> 172.20.56.73 TCP 60 1945â+'80 [<None>] Seq=1 Win=512 Len=0
 21 188.561805 172.20.56.41 -> 172.20.56.73 TCP 60 1946â+'80 [<None>] Seq=1 Win=512 Len=0
 22 188.561873 172.20.56.41 -> 172.20.56.73 TCP 60 1947â+'80 [<None>] Seq=1 Win=512 Len=0
 23 188.561939 172.20.56.41 -> 172.20.56.73 TCP 60 1948â+'80 [<None>] Seq=1 Win=512 Len=0
 24 188.562005 172.20.56.41 -> 172.20.56.73 TCP 60 1949â+'80 [<None>] Seq=1 Win=512 Len=0
 25 188.562073 172.20.56.41 -> 172.20.56.73 TCP 60 1950â+'80 [<None>] Seq=1 Win=512 Len=0
 26 188.562140 172.20.56.41 -> 172.20.56.73 TCP 60 1951â+'80 [<None>] Seq=1 Win=512 Len=0
 27 188.562207 172.20.56.41 -> 172.20.56.73 TCP 60 1952â+'80 [<None>] Seq=1 Win=512 Len=0
 28 188.562275 172.20.56.41 -> 172.20.56.73 TCP 60 1953â+'80 [<None>] Seq=1 Win=512 Len=0
 29 188.562342 172.20.56.41 -> 172.20.56.73 TCP 60 1954â+'80 [<None>] Seq=1 Win=512 Len=0
 30 188.562409 172.20.56.41 -> 172.20.56.73 TCP 60 1955â+'80 [<None>] Seq=1 Win=512 Len=0
 31 188.562476 172.20.56.41 -> 172.20.56.73 TCP 60 1956â+'80 [<None>] Seq=1 Win=512 Len=0
 32 188.562543 172.20.56.41 -> 172.20.56.73 TCP 60 1957â+'80 [<None>] Seq=1 Win=512 Len=0
 33 188.562610 172.20.56.41 -> 172.20.56.73 TCP 60 1958â+'80 [<None>] Seq=1 Win=512 Len=0
 34 188.562678 172.20.56.41 -> 172.20.56.73 TCP 60 1959â+'80 [<None>] Seq=1 Win=512 Len=0
```

Figure 4-8: Sample of assembled packets by hping3

After the launch of the DoS from the attacking node to the cloud server, the tcpdump captured the raw attacking traffic ingress to the cloud coming from the attacking machine. The collected raw attacking traffic was saved as a pcap file. The pcap file had 888 transmission records. This dataset was used to test the Bayesian network classifiers. A sample of the raw tcpdump is shown in Figure 4-8.

4.2.3.1 Data Analysis using Wireshark

To perform a detailed analysis of the tcpdump output presented in the previous sub-section, we used Wireshark tool which was installed on the server side. Wireshark is GUI version of tcpdump and it has the capability of giving more detailed information of the tcpdump capture file. As illustrated from the sample shown in Figure 4-8, we can see that a TCP packet was sent from a host with IP address 172.20.56.41 and received by destination host with IP address 172.20.56.73 at a particular time. Wireshark tool was used to mine more information that can be

revealed on the Wireshark GUI. Wireshark transforms the output from packet-level data to connection-level data.

The analysis of the dataset was conducted by applying the Bayesian network classifier under study to classify the network instances as either normal or DoS traffic.

4.2.4 Data pre-processing

In building Bayesian network models, data pre-processing is an important step for achieving the best performance (Kotsiantis, Kanellopoulos & Pintelas, 2006). Generally, data preprocessing is a technique for converting raw data into understandable or usable format. Real-world data mostly comes incomplete, inconsistent and containing various errors. Data preprocessing is a method that resolves issues of this nature. In the data pre-processing stage, the dataset collected from *tcpdump* contained packet header information. It is a violation of standard procedure to have training dataset in its raw format. The data must be subject to standard pre-processing procedure which includes steps such as data cleaning, integration, feature construction to derive new higher-level features, feature selection to choose the optimal subset of relevant features, reduction and discretization (Kotsiantis, Kanellopoulos & Pintelas, 2006). The product of this stage will be a training dataset.

According to (Bhattacharyya & Kalita, 2014), pre-processing of the data prior training of a BN is important for the following reasons:

a) It reduces feature dimensionality and helps improve the performance of the BN algorithm.
b) It removes the irrelevant or noisy instances in the dataset.
c) It enhances the data quality thus consequently improving the performance of the learning mechanism.
d) The resultant model will be more accurate.

A number of pre-processing methods and algorithms have been used by various authors. Before the algorithms are applied, the data should be prepared accordingly and it should be in the appropriate format suit for training a Bayesian network. Prior the data pre-processing phase, the *tcpdump* data contained packet header information as shown in Figure 4-8:

- Flags such as SYN, FIN, PUSH, RST, URG flags;
- Data sequence number of each packet;
- Data sequence number of the return packet;
- Size of the available receive buffer in bytes;
- Length of the packet;

As discussed in Section 4.2.2, the NSL-KDD dataset has 41 features in connection-level. Therefore, the live packet capture must hence be converted to connection-level format. There are basically two ways found in literature for this purpose. First one uses the Bro language (Lee et al., 1999) and the other uses the a software known as tcptrace (Ostermann, 2003). The aforementioned methods are capable of extracting the 41 features from raw tcpdump data. During data pre-processing of the live packet capture, we encountered a major limitation in the extraction of all the features found in the NSL-KDD dataset. This was due to inaccessibility of the specialized Bro programs to perform the extraction as well as the challenges encountered in using tcptrace. To mitigate this challenge, we used (Palmer, 2011)'s notion of using Wireshark for pre-processing the packet capture so that they could be used with NSL-KDD. However, Wireshark only managed to extract on five features that are displayed in Table 4-1. This meant that the original NSL-KDD dataset be reduced down to the tabulated features and this was accomplished by using the perl script found in WEKA.

Table 4-1: Features extracted using Wireshark

Number	Parameter
1	Duration
2	Protocol type
3	Service type
4	Src_bytes
5	Dst_bytes

4.2.5 Bayesian Networks Classifier Model

The Bayesian networks classifiers were implemented in the WEKA machine learning workbench. WEKA provides a comprehensive collection of machine learning algorithms and

data processing (Hall et al., 2009). WEKA framework can perform classification and regression tasks. It also allows for:

- Model learning,
- Testing and evaluating trained network.

4.2.5.1 Induction of a Bayesian network classifier

The induction of a Bayesian network classifier involves the following steps taken:

- Importation of the training set
  - The following three processes were carried out prior training:
    i. We used the filter of *ReplaceMissingValues* within WEKA to make sure missing values were replaced.
    ii. We used the appropriate discretization filter in WEKA to discretise numeric attributes.
    iii. Class imbalance in the training dataset was mitigated using the filter of *SMOTE*. This is for ensuring that there is a right distribution to evade bias towards the majority class.

4.2.6 Performance evaluation

The classification performance of the Bayesian network classifiers under study was examined during the testing stage, where the models were tested on a dataset with DoS attacks. The classification ability was measured by measuring the predictive accuracy. In this study, binary classification is applied. The confusion matrix was the most appropriate way to present the binary classification results. In the binary classification nature of detection, there are the following four possibilities:

True Positive (TP): the model correctly classified an intrusion as intrusive.

True Negative (TN): the model correctly classified a normal event as normal.

False Positive (FP): the model incorrectly classified a normal event as intrusive.

False Negative (FN): the model incorrectly classified as intrusive behaviour as normal

A graphical format is used to visualize how the aforementioned concepts are related in a confusion matrix. The confusion matrix is depicted in Table 4-2.

Table 4-2: Confusion matrix for binary classification model

Actual	Predicted	
	Attack	Normal
Attack	True Positive (TP)	False Negative (FN)
Normal	False Positive (FP)	True Negative (TN)

The classification problem can be solved by further processing information extracted from the confusion matrix. Listed below is the performance measure that was adapted in this study.

- Accuracy: this measure indicates the total number of events that are correctly classified including normal and intrusive events (See Equation 4.1)

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+FP+TN+FN}} \quad \textbf{(4.1)}$$

Accuracy is the simplest form of evaluation in classifier performance. Flores et al. (2011, p. 378) provided the motivation for the researcher to evaluate the performance in terms of accuracy when they said, "*Accuracy is an important factor when evaluating a classifier or even when studying more general data mining techniques".* Researchers such as (Friedman & Goldszmidt, 1996), (Madden, 2009), (Flores et al., 2011) used accuracy to measure the classification performance of Bayesian network classifiers.

4.2.7 Results analysis

To provide a descriptive and sound comparison between the results obtained from the empirical study, we considered the objects being investigated (model complexity, training sample size, discretization approaches and score function) and graphically represented them by mapping the

classification accuracy values. The information represented is highly useful, since we were comparing different Bayesian network classifiers the results obtained were bound not to be commensurable. Therefore we conducted some statistical tests to provide a different dimension for analysis.

For this reason we employed the Friedman test based on the guidelines provided in (Demˇsar, 2006). We are interested in how the classification performances of classifiers are conditioned by structural complexity, size of training set, different discretization approaches and score functions. Friedman test is one of the non-parametric statistical tests and works in a similar manner as ANOVA. To compare the factors against each other we use the Nemenyi test and Turkey test for Friedman test and ANOVA respectively. In experiments where the classifier performance is conditioned by two factors e.g. training sample size and discretization approach used, we made box plots for visual comparison. An example of a box plot is shown in Figure 4-9.

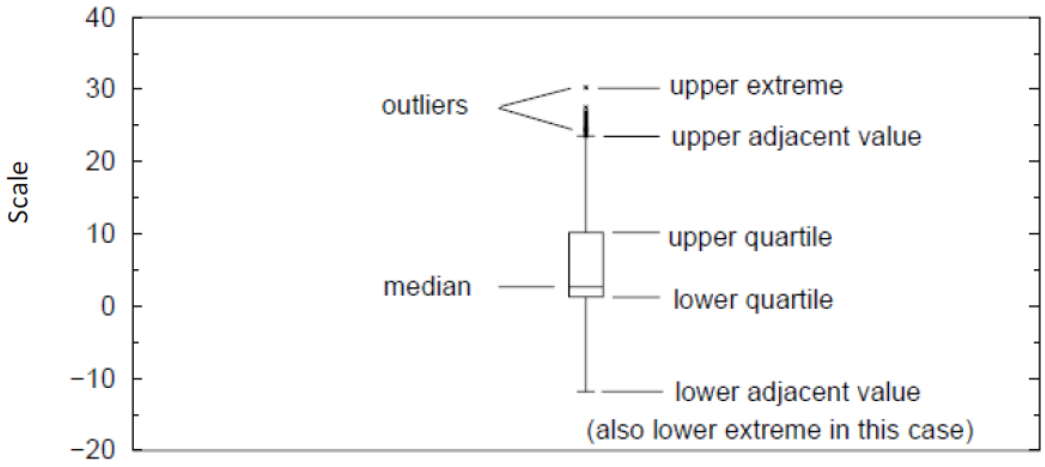


Figure 4-9: Box plot parameters

4.2.8 Experiments set up

Figure 4-2 depicts the framework used for our experiments. We obtained the results for the classification of DoS attacks in a cloud platform using the experiment with the following configurations comprising of six discretization approaches; EW, EF, MDL, KON-MDL, CAIM and ur-CAIM, five score functions; BDeu, AIC, MDL and BAYES, five classifiers NB, TAN, AODE and GBN, varying training set size. We have model combinations that were implemented in WEKA workbench. Table 4-3 presents characteristics of the dataset used.

Table 4-3: Data characteristics

Dataset	Instances	Attributes	Discrete	Continuous	Classes
Training	1000	5	2	3	2
Testing	880	5	2	3	
Total	1880				

The GBN structures were trained with K2 and HC learning algorithms provided in WEKA setting the maximum number of parent node as three. All the Bayesian network classifiers were implemented within the Weka tool. We adopted the widely acceptable method that produces statistically meaningful results, the 10-fold cross-validation for the training purposes. 10 fold cross-validation randomly partitions the available data into 10 disjoint subsets of equal sizes. 9 sets are used for constructing the classifiers and the remaining set used as the test set for estimating the accuracy of the classifiers. The process is repeated 10 times until each partition is used as a test set once. Throughout the experiments, the cross-validation folds remained invariant. Runs with the four classifiers were conducted using the same training set. A two-tailed test with 0.95 confidence level was applied to measure significance so as to compare the classifiers.

4.3 Conclusion

This chapter has outlined the methods used and data techniques implemented in our study. The conceptual framework followed for the experiments was presented. A section was devoted for the description of the intrusion dataset used as our training set. The cloud environment was in which the empirical study was conducted was described as well as the data collection and pre-processing techniques applied. Hardware and software requirements, specifications and the methodology were also presented. The test-bed setup and the steps for the experiments conducted in the study were clearly presented. This chapter also stated the performance evaluation criteria used to assess the performance of the Bayesian network classifiers. Chapter 5 presents the results of the experiments done and their respective discussions and findings.

Chapter 5: Results and Discussion

This chapter presents and discusses the results from the empirical experiments described in Chapter five. The Chapter is sectioned as follows, section 5.1 presents the results of the various experiments conducted, section 5.2 provides discussions and statistical analyses of the results of the experiments and lastly 5.3 concludes the chapter. By presenting, interpreting and analysing a series of graphs from the experiments the chapter explores how the classification performance of Bayesian network classifiers, empirically calculated using Accuracy on the test instances, is affected by model complexity, the sample size of the training set, the type of discretization approach and the type of score functions. In the chapter we further do perform tests such as Friedman tests, ANOVA and Nemenyi so as to deduce sound conclusions. The experiments were conducted with respect to the following series of steps;

- *Individual factor tests: - investigation of how each factor individual impact the classifier performance with all the other factors held invariant.*
- *Collective factor test: - investigations of how several combinations of training parameters collectively affect classification performance.*

5.1 The results

5.1.1 Model Complexity

In the first experiment, we focussed on how model complexity of Bayesian network classifiers, that entails the progression from the simple model NB through AODE and lastly GBN, affects the predictive power of the classifier when subjected to a test set. Literature has revealed that the augmentation of the NB model is not commensurate with classification performance (Cheng & Greiner, 1999; Madden, 2009). Therefore our empirical study investigates how the progression from NB to GBN affects the classification of DoS attacks. Our quest was to identify the classifier that will exhibit the best classification performance in the detection task. The GBN classifier was trained with both K2 and HC search algorithms just for empirical curiosity. With all the other parameters invariant, the training set size at 1000 instances and the test set at 880, PKID was used as the discretization approach and MDL was chosen as the scoring function), the results for the experiment performed are shown in Table 5-1.

Table 5-1: Model complexity and classification accuracy

Model	Accuracy
NB	74.6747
TAN	86.8969
AODE	89.683
GBN-K2	86.7968
GBN-HC	85.996

5.1.2 Sample size

Of the 1880 samples used in this research, 1000 were dedicated to training. If the classification performance remains invariant while training set is reduced then it is possible to curb the duration of the training session and computational costs (Brain & Webb, 1999). In this second set of experiments, we studied the impact of the training sample size on the classification performance of Bayesian network classifiers. The five classes of Bayesian networks classifiers were inducted to distinguish between normal and DoS traffic using either the full training sample set of 1000 instances or the reduced variations of the training set. We partitioned the full training dataset into ten subsets. These subsets were generated at 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100% of the full designated training set of 1000 instances. As alluded by (Bielza & Larrañaga, 2014), learning a Bayesian network from data implies that the conditional probabilities will be physical. It is that end that the size of the training set is crucial in inducting accurate probability estimates. Therefore we chose to start to generate the training set size at 10% of the total training sample size. The results for the experiment conducted are shown in Table 5-2. Figure 5-1 presents the results graphically.

Table 5-2: Varying training set size

	NB	TAN	AODE	GBN-K2	GBN-HC
10%	74.5721	77.7477	78.0616	75.9468	74.8097
20%	82.9648	83.9698	83.9698	84.4724	82.4623
30%	83.9799	84.6488	84.3144	84.6488	87.3244
40%	84.9875	85.7859	86.7419	85.2381	84.7368
50%	82.5852	82.9864	87.7956	82.7856	84.99
60%	84.4908	85.8264	87.4958	85.8264	84.9917
70%	86.9914	87.4212	88.4241	87.4212	86.5616

80%	87.1214	87.2466	88.373	87.4969	86.2453
90%	87.3274	88.1069	88.441	87.9955	86.5479
100%	86.6747	86.8969	89.683	86.7968	85.996

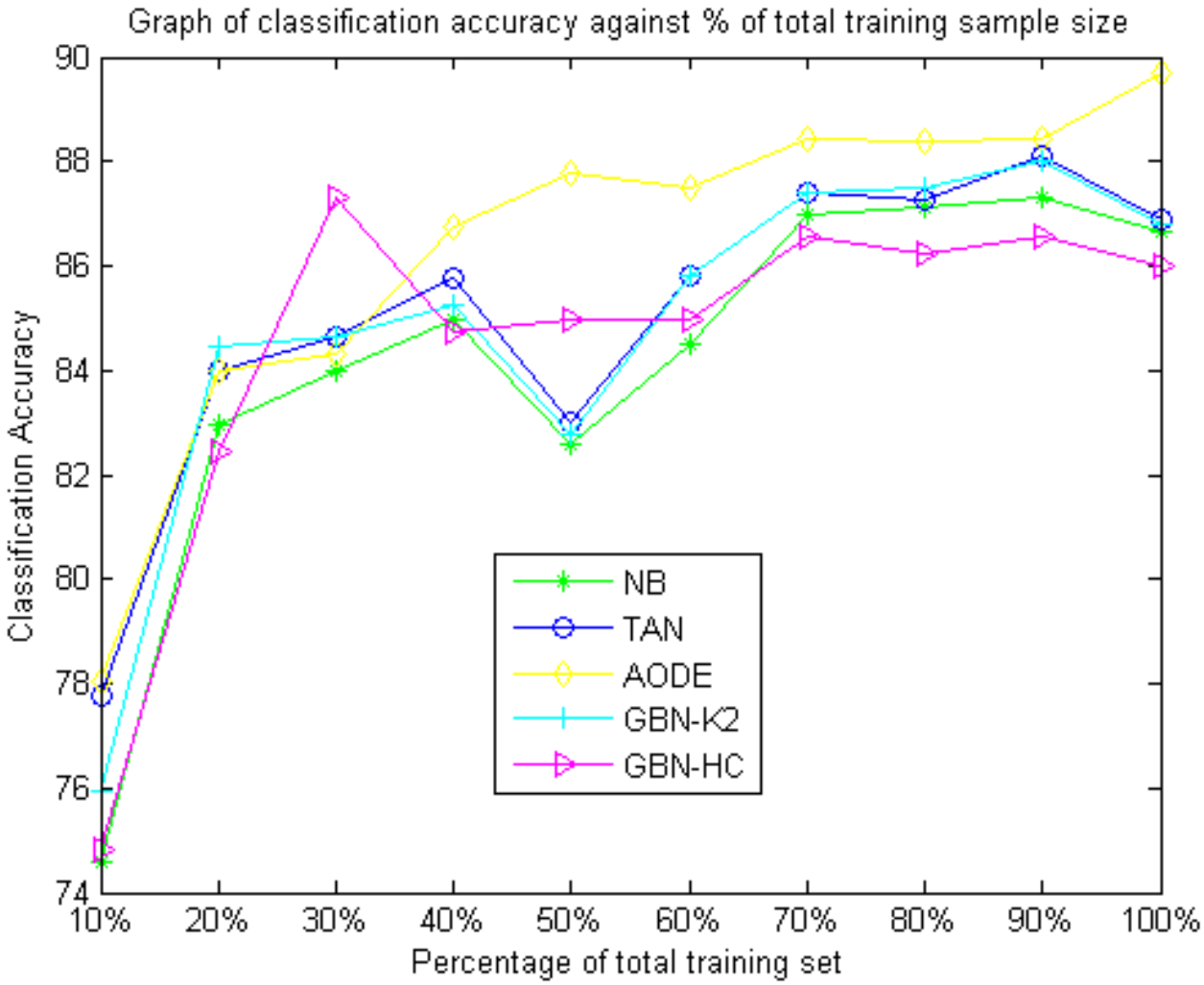


Figure 5-1: Varying training set size

Appendix C shows a boxplot that summarises the response of Bayesian network classifiers to changes in the training sample size. Friedman test was applied to compare the different classifiers for each training sample size. There was a statistically significant difference in classification performance of the Bayesian network classifiers over varying training sample size, $_{\chi}^{2}(4) = 20.154$, $p = 0.0004657$. Friedman's test claims there is a statistical difference between the performances of the classifiers hence it was meaningful to conduct multiple comparisons so

as to identify differences between the classifiers. For the output of Friedman is shown in Appendix A. Pairwise comparisons using Nemenyi post-hoc test were performed. The output of the test is shown in Appendix B. According to the Nemenyi post-hoc test for the Bayesian network classifiers comparison, NB differs highly significantly ($p < 0.05$) to AODE and GBN-K2. The remaining comparisons are not significant ($p > 0.05$).

In our first set of empirical study, we examined the classification performance of five classes of the Bayesian network classifiers. During the experiment, a fixed training sample size was used as well as a similar discretization approach and the score function was chosen for the induction process. It is the model complexity that varied. In this second scenario of experiments, the sample size, which is the number of training instances, is varied whilst the other configurations are fixed.

5.1.3 Discretization

There exist different types of discretization algorithms; some are supervised, others unsupervised, some need user input while others are automatic, some provide improved experiment runtime. In this third scenario, 6 discretization approaches were evaluated primarily on how they impact the classification performance of Bayesian networks classifiers, further experiments were conducted to assess how a collective of a discretization and training sample size impact the classification performance of the Bayesian network classifiers. The Equal Width (EW), Equal Frequency (EF), Proportional k-interval (PKI), Fayad and Irani's Entropy based Minimum Description Length (MDL), Class Attribute Interdependency Maximization (CAIM) and ur-Class Attribute Interdependency Maximization (ur-CAIM) were considered in the task of detecting DoS attacks.

In the first set of the empirical study, the five classes of Bayesian networks classifiers were trained on the same parameters such as a fixed training set and score metric. The performance of the Bayesian networks classifiers was evaluated on the 888 instances of test data. Score function used is MDL. Figure 5-2 shows the classification accuracy with the different algorithms and Table 5-3 shows the results of the classification accuracy on both train and test dataset for the experiments.

Table 5-3: Results of discretization approach comparison

Classifier	EW5	EW10	EF5	EF10	PKID	MDL	CAIM	ur-CAIM
NB	74.9	76.4	75.2	79.5	83.8	82.0	79.6	81.4
TAN	76.7	77.2	76.9	77.7	85.6	87.7	82.6	84.7
AODE	79.1	80.7	83.8	81.7	87.5	87.5	85.8	86.2
GBN-K2	78.3	79.9	82.3	80.6	86.1	87.1	84.0	83.6
GBN-HC	77.5	78.0	81.2	80.2	85.3	86.6	82.9	84.2

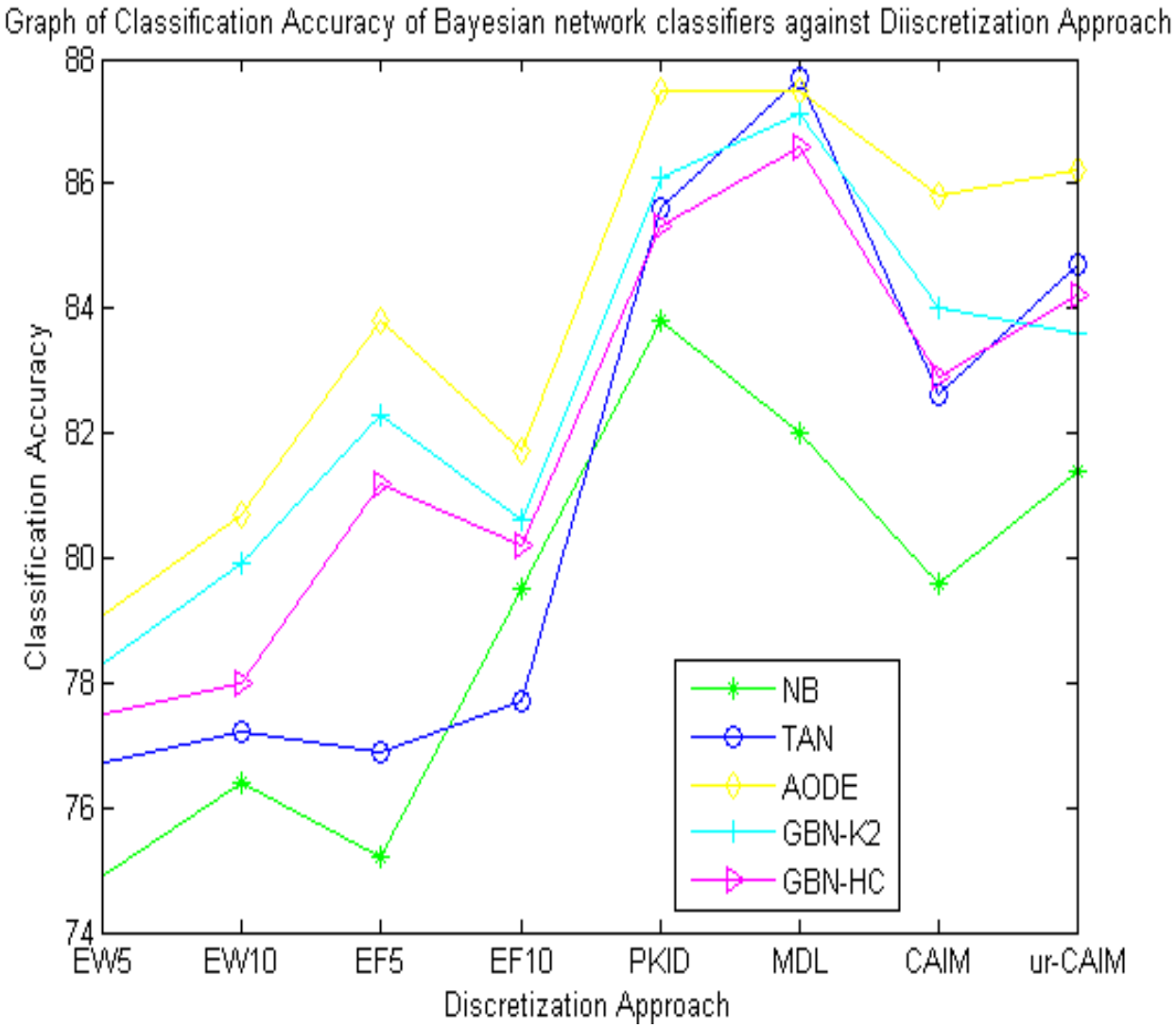


Figure 5-2: Discretization approach against Classification performance

Appendix F shows a boxplot that summarises the response of Classification accuracy to differing discretization approaches used. According to the Friedman test shown in Appendix D there was a statistically significant difference in classification performance of the Bayesian network classifiers over different discretization approach employed $\chi^2(7)$ = 33.56, p = 2.082e-05. Friedman's test claims there is a statistical difference between the performances of the discretization methods hence it was meaningful to conduct multiple comparisons so as to identify differences between the classifiers. Pairwise comparisons using Nemenyi post-hoc test were performed. The output of the test is shown in Appendix E. According to the Nemenyi post-hoc test for the Bayesian network classifiers comparison, the modelled using EW differs highly significantly ($p < 0.05$) to classifier trained using MDL and PKID. The remaining comparisons are not significant ($p > 0.05$).

Appendix I shows a boxplot that summarises the response of Classification accuracy to differing discretization approaches used. According to the Friedman test shown in Appendix G there was a statistically significant difference in classification performance of the Bayesian network classifiers over different discretization approach employed $\chi^2(4)$ = 10.13, p = 0.03833. Friedman's test claims there is a statistical difference between the performances of the discretization methods hence it was meaningful to conduct multiple comparisons so as to identify differences between the classifiers. Pairwise comparisons using Nemenyi post-hoc test were performed. The output of the test is shown in Appendix H.

The second set of experiments investigates the behavior of the discretization approach as training sample size increases for each Bayesian classifier.

NB

Table 5-4: Results of discretization against size of training set of NB

	EW5	EW10	EF5	EF10	PKID	MDL	CAIM	ur-CAIM
10%	79.899	81.9192	82.9293	83.0515	82.9293	85.9596	84.9495	82.9293
20%	79.9497	78.9447	83.4673	82.9477	84.4724	85.9799	81.4573	84.4724
30%	80.301	79.6321	85.3177	82.3077	86.3211	86.6555	82.9766	85.9866
40%	78.7218	80.4762	84.4862	83.2331	85.7393	87.2431	83.7343	84.7368
50%	77.5752	81.1824	84.1884	82.7856	85.5912	87.5952	82.986	85.7916
60%	78.6477	78.9816	84.1569	82.8214	85.3255	87.4958	82.6544	85.4925
70%	78.6981	79.9857	84.2775	82.8469	85.8512	87.568	82.8469	85.422
80%	78.9088	80.2378	84.2428	82.7409	85.8698	87.3717	82.4906	85.4944
90%	79.1892	80.2378	83.9933	82.5473	86.218	87.3304	82.3248	85.3281
100%	77.8879	80.3904	83.3934	83.1932	85.7958	87.3974	82.6927	84.8949

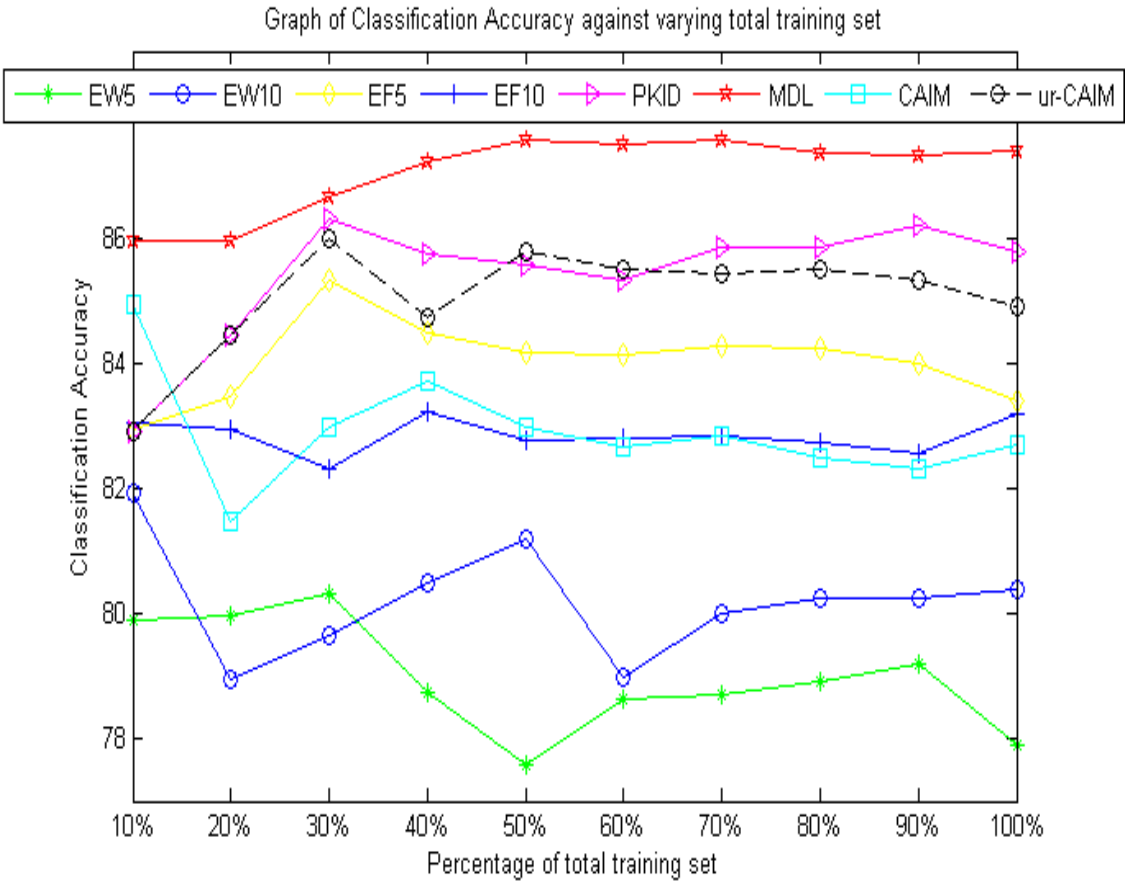


Figure 5-3: Classification performance versus training size

TAN

Table 5-5: Results of discretization approach against size of sample set TAN

	EW5	EW10	EF5	EF10	PKID	MDL	CAIM	ur-CAIM
10%	79.899	81.9192	81.9192	84.9495	82.9293	86.9697	82.9293	85.3954
20%	79.9497	78.9447	82.4623	82.4623	85.4774	85.9799	82.4623	83.9698
30%	78.2943	78.6288	84.6488	82.6421	86.3211	86.99	82.9766	85.6522
40%	78.4712	79.4737	84.7368	83.7343	85.99	87.4937	83.2331	84.9875
50%	77.976	78.978	83.988	82.7856	85.5912	87.3948	83.3868	85.992
60%	78.9816	78.9816	84.1569	82.8214	85.8264	87.4958	82.3205	86.4942
70%	78.9843	79.9857	84.4206	82.99	86.1373	87.568	82.1316	86.4235
80%	79.2665	80.2378	83.7422	85.3692	86.3705	87.1214	81.99	86.1202
90%	79.5072	80.1001	83.7709	87.2191	86.4405	87.2191	82.3248	85.5506
100%	78.0881	80.5906	83.4935	85.5956	86.5966	87.6977	82.5926	85.3954

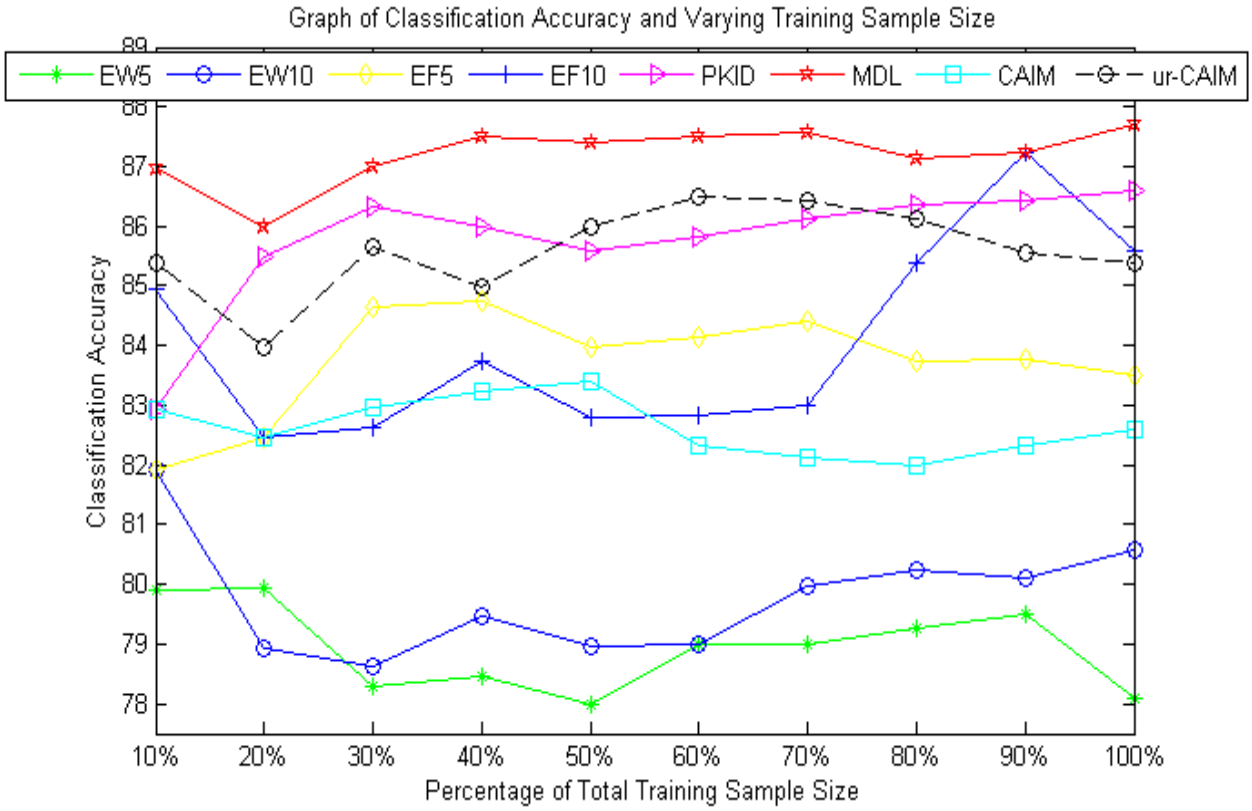


Figure 5-4: Classification accuracy versus training set

AODE

Table 5-6: Results of discretization approach against training set size AODE

	EW5	EW10	EF5	EF10	PKID	MDL	CAIM	ur-CAIM
10%	79.899	81.9192	82.9293	84.9495	82.9293	86.9697	84.9495	82.9293
20%	79.9497	78.9447	83.4673	81.4573	84.9749	85.4774	82.4623	83.9698
30%	80.6355	80.6355	85.3177	85.3177	87.3244	86.6555	82.6421	84.6488
40%	80.2256	81.4787	84.7368	85.4887	86.9925	87.2431	82.9825	84.9875
50%	78.5772	81.1824	83.988	85.7916	87.1944	86.994	82.3848	85.992
60%	78.9816	80.6511	83.823	85.6594	87.3289	86.995	82.1536	84.9917
70%	78.9843	81.1302	84.1345	86.7096	87.568	87.4249	83.5622	85.5651
80%	79.9821	80.2378	83.617	86.1202	87.622	87.3717	83.2416	85.1189
90%	79.9841	80.1001	83.1034	86.4405	87.7753	87.2191	83.1034	85.4394
100%	78.0881	80.4905	83.2933	86.6967	87.4975	87.7978	83.2933	84.995

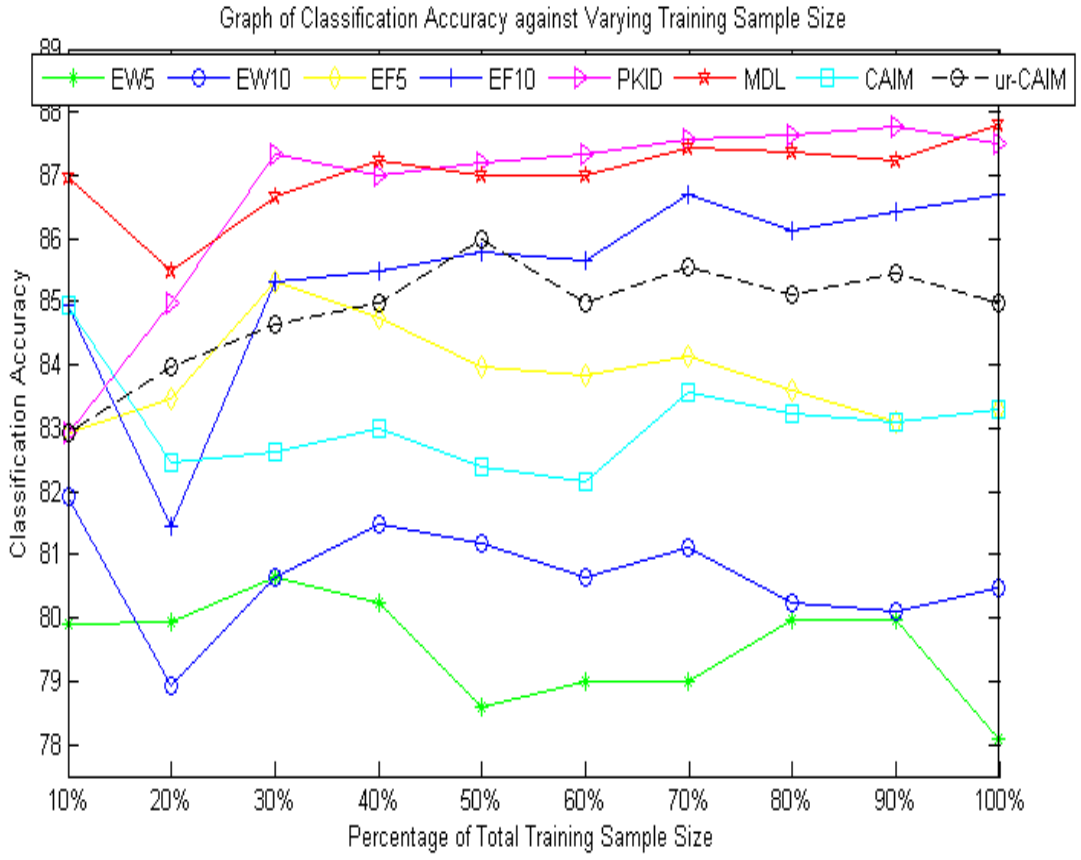


Figure 5-5: Classification accuracy versus training set

GBN-K2

Table 5-7: Results of discretization approach against training set size

	EW5	EW10	EF5	EF10	PKID	MDL	CAIM	ur-CAIM
10%	79.899	81.9192	81.9192	84.9495	82.9293	86.9697	83.9394	80.9091
20%	79.9497	78.9447	83.4673	81.4573	85.4774	85.9799	81.9598	84.9749
30%	78.2943	78.2943	85.3177	82.6421	86.3211	86.6555	83.6455	85.9866
40%	78.2206	79.4737	83.2331	83.4837	85.99	87.4937	82.9825	85.4887
50%	77.7756	78.978	83.7876	82.7856	85.5912	87.5952	82.5852	85.5912
60%	78.9816	78.9816	84.4908	82.8214	85.8264	87.4958	82.3205	85.3255
70%	78.9843	79.9857	84.4206	83.133	86.1373	87.568	82.5608	85.1359
80%	79.2665	80.2378	83.8673	82.7409	86.3705	87.3717	82.3655	85.2441
90%	79.5072	80.1001	83.9933	82.5473	86.3293	87.1079	82.2136	85.3281
100%	77.988	80.5906	83.3934	82.5473	86.0961	87.6977	82.2136	84.4945

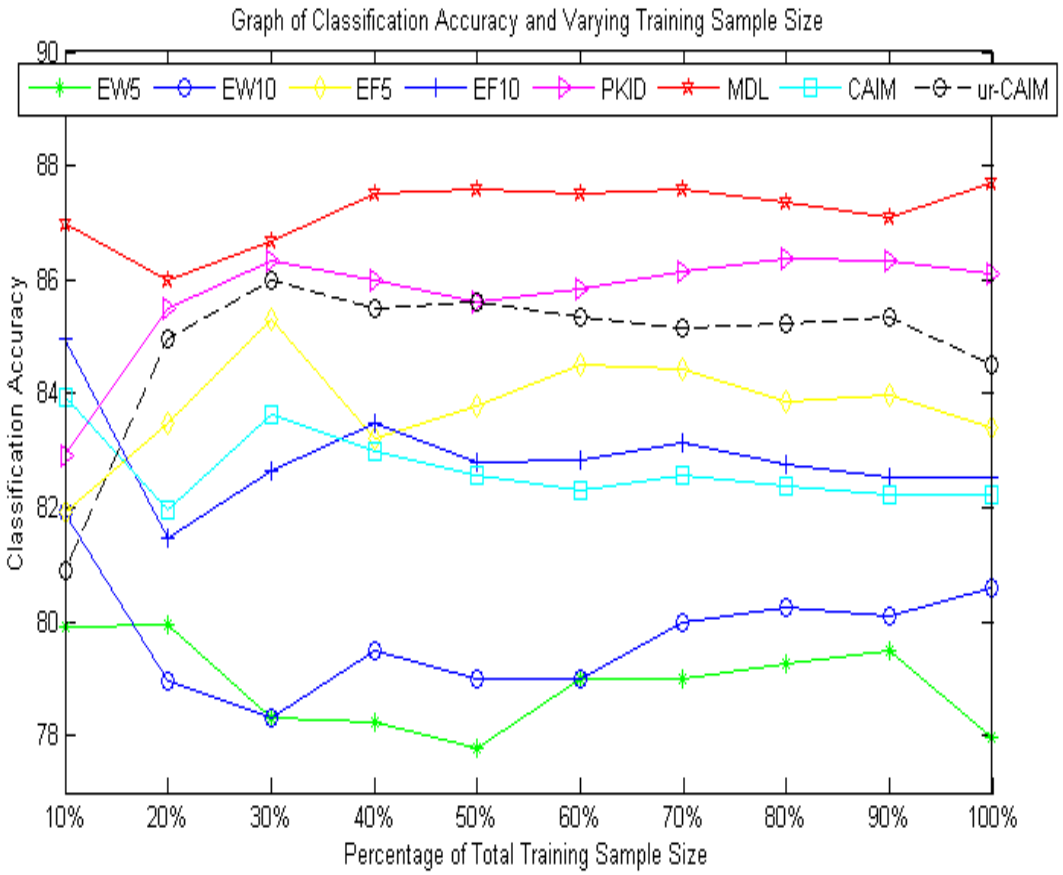


Figure 5-6: Classification accuracy against training set size

GBN-HC

Table 5-8: Results of discretization approach against training set size GBN-HC

	EW5	EW10	EF5	EF10	PKID	MDL	CAIM	ur-CAIM
10%	69.596	70.6061	77.8788	79.899	79.596	86.9697	82.9293	84.9495
20%	69.799	72.8141	81.9598	80.9548	81.4573	84.9749	82.4623	82.9648
30%	70.2007	70.2007	80.9699	83.9799	84.6488	86.3211	83.311	84.9833
40%	69.8997	71.6541	83.985	84.2356	84.4862	86.9925	82.7318	85.7393
50%	69.1182	72.9259	83.3868	85.1904	85.7916	87.1944	81.7836	86.7936
60%	70.2671	73.4391	82.8214	83.823	84.9917	87.1619	82.6544	86.4942
70%	77.9828	80.1001	83.8484	83.9914	85.8512	87.4249	82.8469	85.1359
80%	72.4329	76.6083	83.7422	82.7409	86.2453	87.2466	82.2403	85.2441
90%	74.2289	76.7631	82.7697	82.5473	85.8843	86.3293	82.2136	85.2169
100%	77.988	77.7878	82.0921	83.1932	85.2953	86.8969	82.7928	84.3944

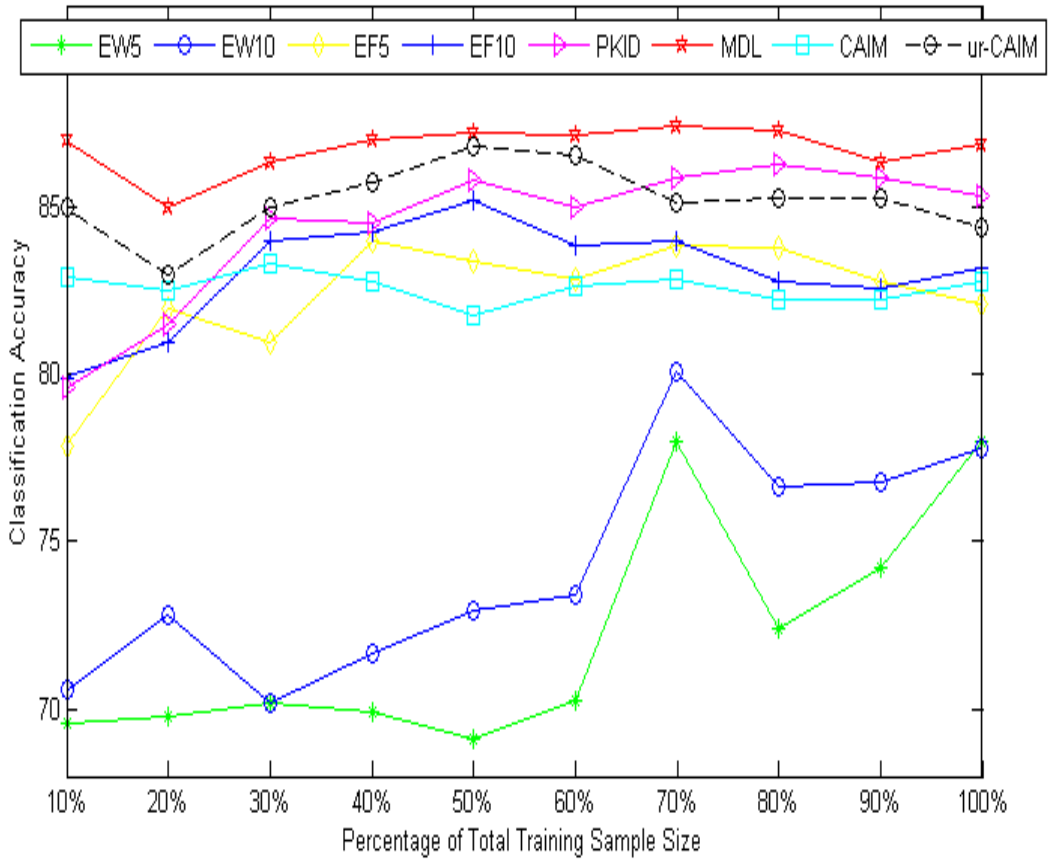


Figure 5-7: Classification accuracy versus training set size

5.1.4 Score Function

There are various types of score functions. They are built under different criterions. The search for the network that maximizes the score is a optimization problem (Liu, Malone & Yuan, 2012). In our experiments, we trained the classifiers on five variables. According to (Liu, Malone & Yuan, 2012), five variables produces 29 281 network structures. In this last scenario, the focus is toward the impact of different scoring functions on the classification ability of Bayesian network classifiers. The Bayes, Minimum description length (MDL), AIC, entropy, BDeu metric were considered in this empirical study. WEKA's default configurations of the score functions were used in the experiments. During the experiments the training sample size remained invariant at 1000 instances and simple discretization approach PKID was used. The results for the experiment conducted are shown in Table 5-9.

Table 5-9: Results of score function on classification performance

Algorithm	**% Correct (Training)**	**% Correct (Testing)**
TAN –S BAYES	88.46	87.998
TAN –S AIC	87.68	86.4965
TAN –S MDL	89.24	86.5966
TAN –S BDeu	87.54	88.1982
GBN K2 –P 3 –S BAYES	88.91	87.7978
GBN K2 –P 3 –S AIC	88.66	86.0961
GBN K2 –P 3 –S MDL	86.13	85.1569
GBN K2 –P 3 –S BDeu	87.34	87.5976
GBN HC –P 3 –S BAYES	86.85	87.7978
GBN HC –P 3 –S AIC	87.04	86.0961
GBN HC –P 3 –S MDL	86.90	85.2953
GBN HC –P 3 –S BDeu	88.67	87.1982

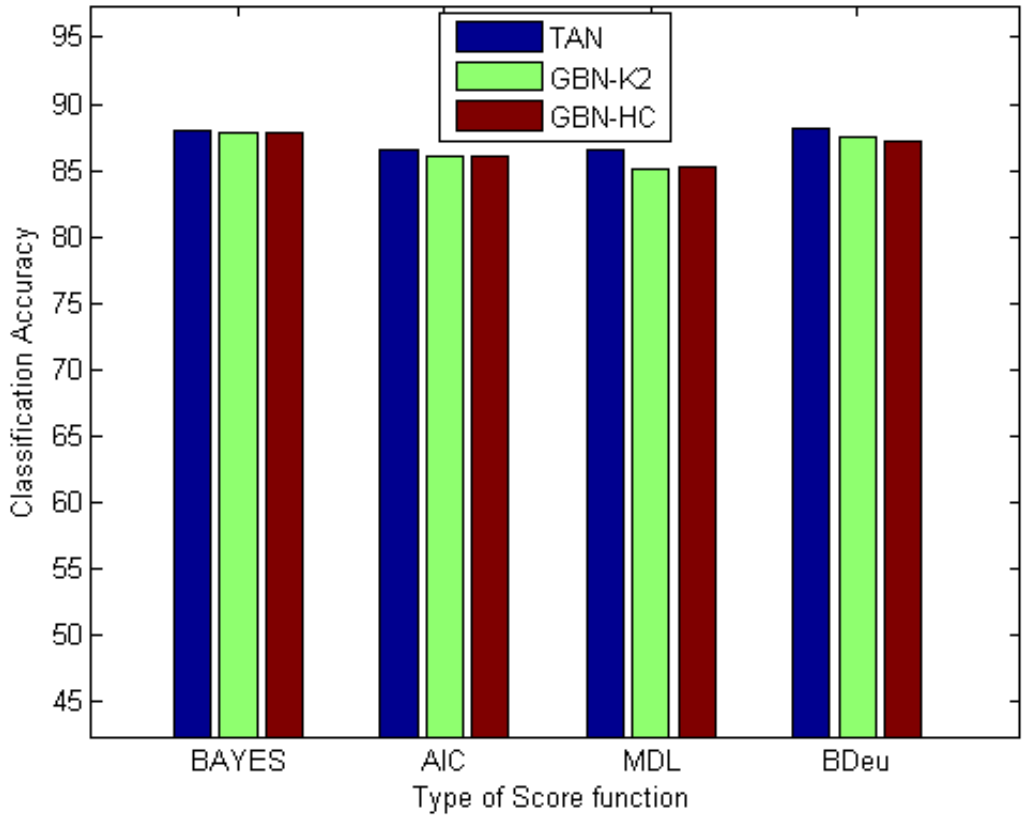


Figure 5-8: Classification accuracy vs type of score function

Appendix M shows a boxplot that summarises the response of Classification accuracy to differing scoring function chosen for the induction process. According to the Friedman test shown in Appendix J there was a statistically significant difference in classification performance of the Bayesian network classifiers over different discretization approach employed $_{\chi}{}^{2}(4) = 10.13$, $p = 0.03833$. Friedman's test claims there is a statistical difference between the performances of the discretization methods hence it was meaningful to conduct multiple comparisons so as to identify differences between the classifiers. Pairwise comparisons using Nemenyi post-hoc test were performed. The output of the test is shown in Appendix K.

In the next set experiments, the impact of training sample size on type of scoring function and ultimately the classification performance is investigated.

5.1.4.1 TAN

Table 5-10: Results of score function against varying training sample set

	BAYES	AIC	MDL	BDeu
10%	70.9091	76.8687	76.8687	77.8788
20%	83.4673	83.9698	83.9698	83.4673
30%	88.3278	84.9833	84.6488	86.3211
40%	86.4912	84.9875	84.9875	86.7419
50%	87.996	82.986	82.986	88.3968
60%	87.6628	85.8264	85.8264	88.3306
70%	88.5673	87.4212	87.4212	88.4241
80%	88.1227	87.4969	87.2466	88.1227
90%	88.3296	88.1069	88.1069	87.1047
100%	87.998	86.4965	86.5966	88.1982

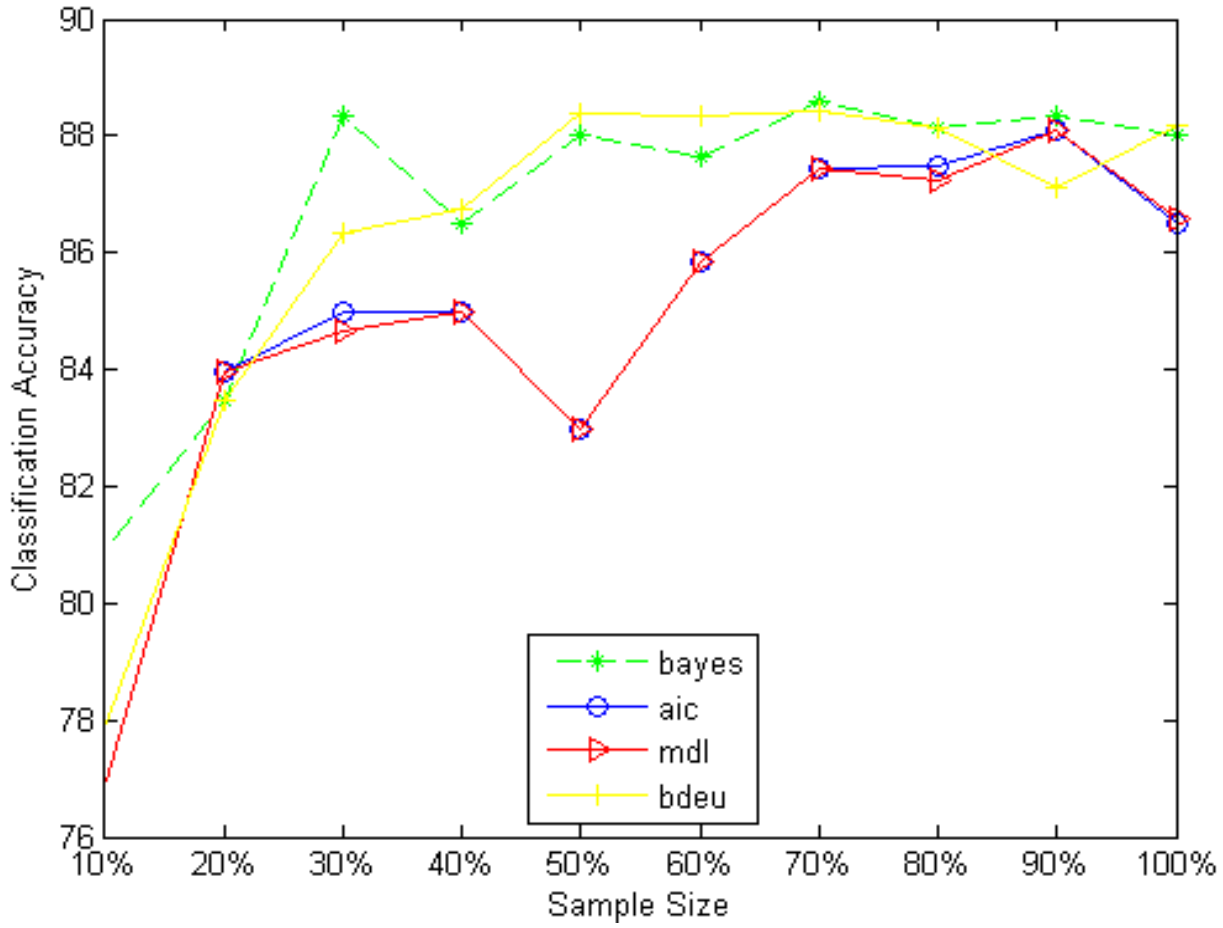


Figure 5-9: Classification accuracy against training sample size

5.1.4.2 GBN-K2

Table 5-11: Results of score function against varying training set size

	BAYES	AIC	MDL	BDeu
10%	80.9091	76.8687	76.8687	76.8687
20%	84.4724	83.9698	83.9698	82.4623
30%	87.3244	84.6488	84.6488	86.6555
40%	87.7444	85.2381	85.2381	86.2406
50%	87.7956	82.986	82.7856	87.1944
60%	88.3306	85.8264	85.8264	87.9967
70%	88.4241	87.4212	87.4212	88.2808
80%	88.373	87.4969	87.4969	88.2478
90%	88.9978	87.9955	87.9955	88.9978
100%	87.7978	86.0961	86.0961	87.5976

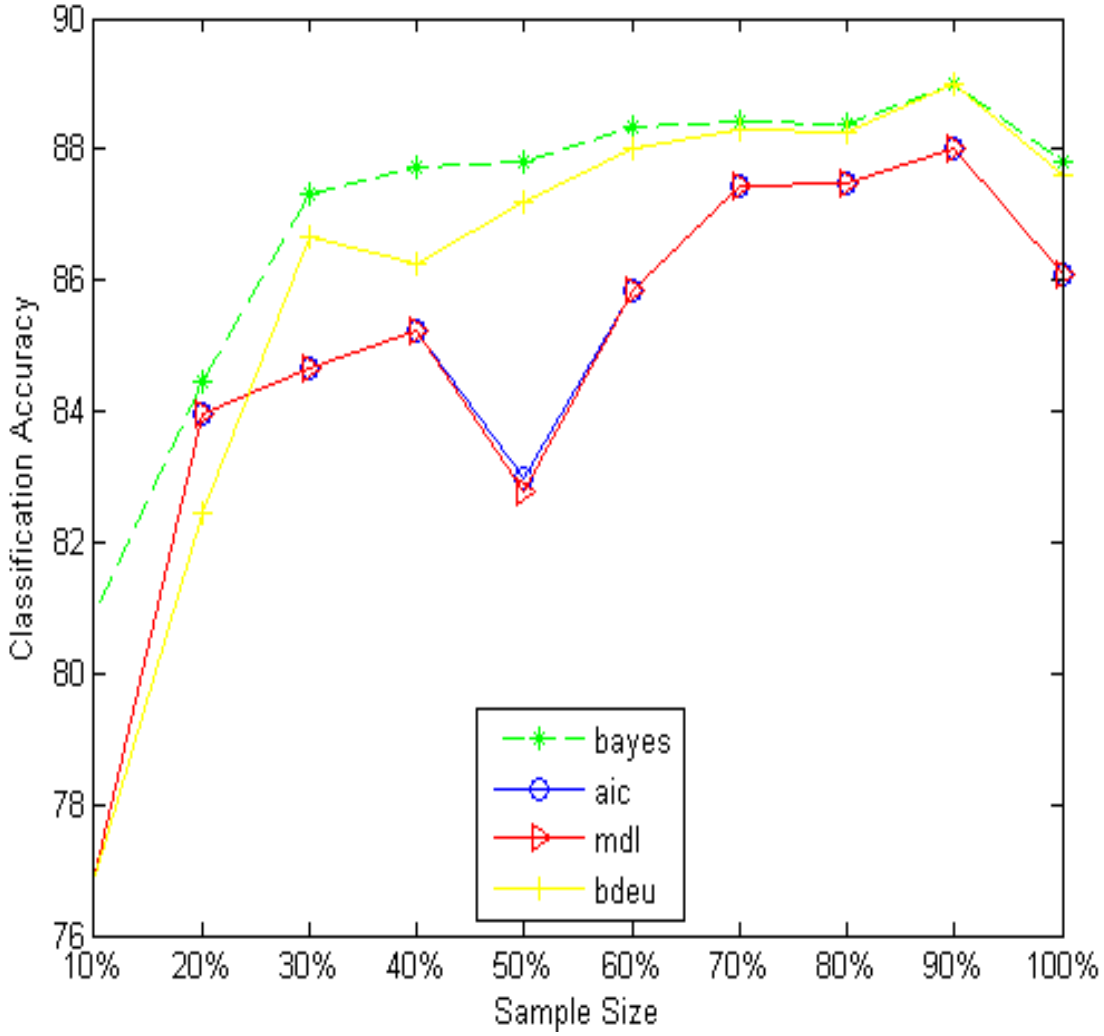


Figure 5-10: Classification accuracy against training sample size

5.1.4.3 GBN-HC

Table 5-12: Results of score function against varying training sample size

	BAYES	AIC	MDL	BDeu
10%	77.8788	79.899	79.899	77.8788
20%	85.4774	84.4724	82.4623	81.4573
30%	87.3244	84.6488	86.3211	85.6522
40%	87.7444	85.2381	84.7368	86.4912
50%	87.7956	82.7856	84.99	87.7956
60%	88.3306	85.8264	84.9917	87.8297
70%	88.4241	87.4212	86.5616	88.4241
80%	88.373	87.4969	86.2453	87.9975
90%	88.8864	87.9955	86.5479	87.8842
100%	87.7978	86.0961	85.2953	87.7978

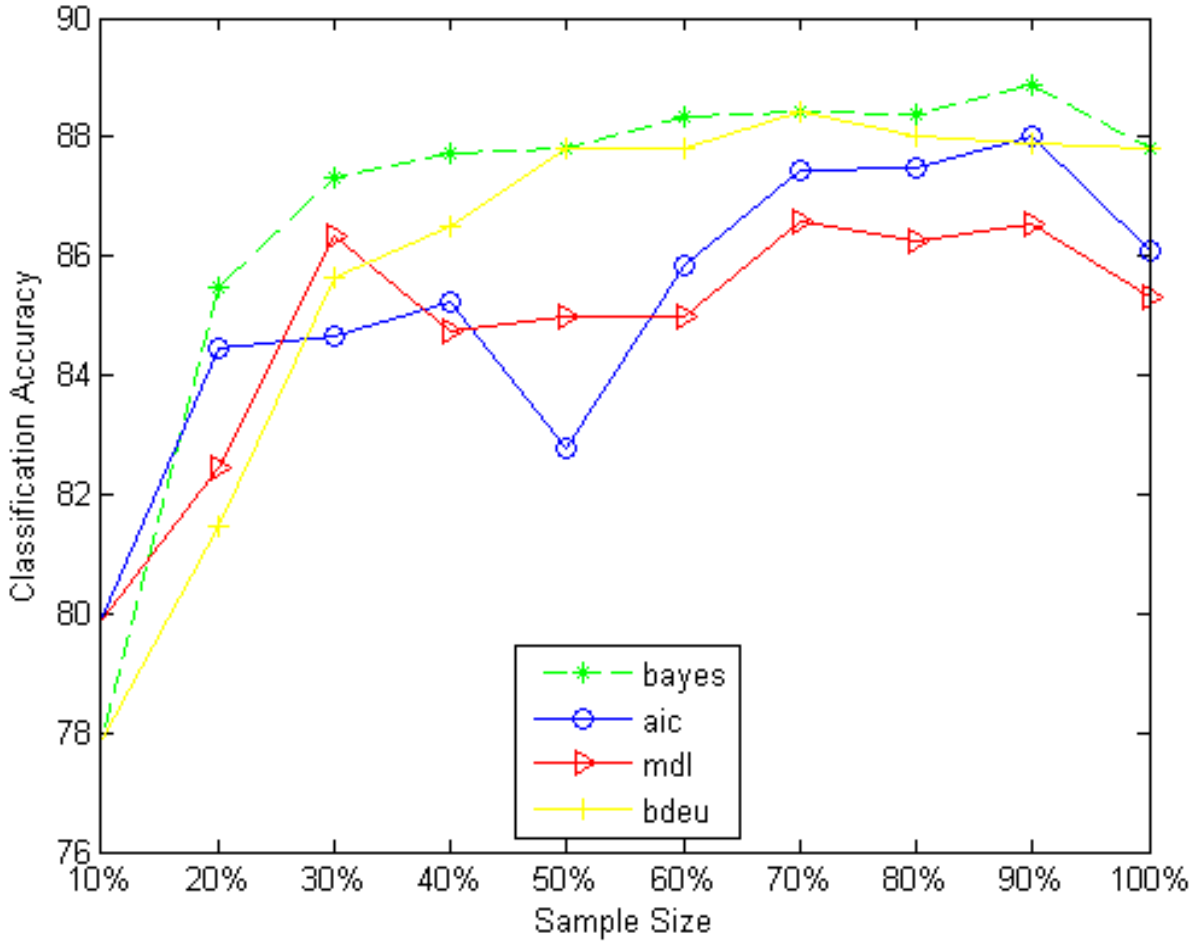


Figure 5-11: Classification accuracy versus training set size

5.2 Discussion of results

This subsection is dedicated to discussing the results presented in section 5.1. The reader should note that the results outlined are based on the effects of a particular parameter under investigation and also how it behaves varied while other parameters remain invariant. For instance, in one of the experiments conducted on the how the choice of discretization approach, various methods were used to assess how they impact the classification performance while the factors such as training sample size, score function remain constant. Furthermore we investigated the behaviour of discretization methods when varying the training sample size. The goal of this empirical study was to exhaust the entire factors under study in how they affect the classification ability of Bayesian network classifiers.

5.2.1 Structural/model complexity

We commence the analysis with the impact of the model complexity on the classification performance of Bayesian network classifiers in detecting DoS attacks in cloud computing platforms. We firstly analyse the results exhibited by the progression in structural complexity starting from the simple NB to the GBN. The results from Table 5-1 show that accuracy values increase from NB through TAN until AODE then the drastically drop when GBN classifiers were used. The results are congruent with the experimental results from (Madden, 2009) and (Friedman, Geiger & Goldszmidt, 1997). The AODE classifier outperformed the NB with the former having an accuracy of 89.63% while the latter has an accuracy of 74.67%. This is due to the fact that the independence assumption among attributes has been relaxed and represented to some degree in AODE. Therefore the class probability estimates of AODE were thus more accurate than those of NB. AODE also outperforms the TAN significantly. The classification accuracy of TAN is 86.90% which lower than that of AODE. This is attributed to the fact that TAN is subjected to computational burden due to model selection and probability estimation. (Cheng & Greiner, 1999) states that model selection tends to increase variance. Thus to evade model selection, AODE pre-empts this phase by averaging the one dependence classifiers. Hence AODE has computational advantage over TAN leading it to have better classification performance. Comparing TAN with NB and also with GBN, TAN outperforms the NB classifier. TAN algorithm also performs better than GBN classifiers while the GBN performs better than NB. GBN-K2 needs a node ordering this may result in additional computational costs. That being said, GBN-HC required no node ordering, however the performance of GBN-K2 exhibited

superior performance than that of GBN-HC. GBN-K2's seemingly algorithmic limitation is important because in specific case where it may be applied, a passable ordering may be derived from domain. TAN has more attribute correlations than NB making it to perform the classification role better. Due to the Markov blanket, the GBN classifier might contain fewer attribute correlations than TAN, thus making TAN a better classifier. (Madden, 2009) reveals that TAN classifier incurs low computational overhead when compared to GBN classifiers.

5.2.2 Training sample size

This section deal with the analysis of how varying training sample sizes affects the classification performance of Bayesian network classifiers in detecting DoS in cloud computing environments. We start our analysis by making reference to Figure 5.1 which depicts how the classification accuracy against the percentage of the total training sample size behaves and changes. In Figure 5.1 it can be vividly observed that the classification accuracy increases as the number of training instances increases. According to Friedman, Geiger & Goldszmidt (1997), a Bayesian network structure that encodes a joint probability distribution $P(X_1, \dots, X_n, C)$ is induced from the training sample. That being said, training sample size plays a critical role in parameter estimation. An increase in training sample size tends to result in an asymptotic approximation for the probability distribution of the real domain under study. Consequently, the larger the training dataset, the lesser the bias. Figure 5.1 depicts that there is generally an improvement in predictive classification performance as we traverse the graph from 10% to 100% of the total training set. The plotted graph is not directly proportional but haphazard as observed by the decrease particularly from 30% to 50%. This can be attributed to uneven distribution of the instances in the training sample. AODE classifier is shown to be the best performing model as compared to its rivals. This can be explained by Webb, Boughton & Wang (2005) who attributed this to more accurate base probability estimates in the one-dependence models. AODE is a high variance model. The performance of the TAN is basically better than the NB, which can be explained to be due to improved probability estimates that come about as a result of the smoothing operation. Our results of the performance of NB with increasing training sample are at variance with those of (Sordo & Zeng, 2005) and (Brain & Webb, 1999) who state that the performance of NB is indistinguishable with changes to training sample size.

5.2.3 Choice of discretization approach

Now we are moving to the empirical study that analyses the impact of the discretization algorithm on the classification performance of Bayesian network classifiers in detecting DoS attacks in cloud computing environments. Figure 5-2 depicts the classification accuracy of the Bayesian classifier under study against the discretization method used prior training. The different line graphs correspond to how the five classifiers behaved for the 8 discretization approaches tested. The discretization methods are represented on the x-axis. It is observed that the classification accuracy followed an irregular exponential curve with each change of discretization approach. The ranking among the classifiers remains unchanged with AODE outperforming the other classifiers yet again except for TAN when discretizing with MDL, we also observe that proportional k-interval discretization is particularly good for AODE whereas EW5 performs worse for almost all the classifiers. Among the supervised discretization algorithms, CAIM performs poorly. Also observed is the least sensitivity of the AODE classifier to the EF10, PKID and MDL discretization approaches applied. In (Flores et al., 2011), the behaviour of Bayesian networks classifiers is examined in terms of bias and variance which the authors termed discretization bias and variance. They studied the analysis extending to some of the classifiers used in this study. The authors pointed out that a sensitive algorithm has a higher variance. Furthermore they discuss about what they referred to as the irreducible error which they described as the level of noise in the data. They used these concepts to explain the number of intervals produced by a discretization approach. A discretization process that results in a large number of intervals tends to result in low bias models. This therefore explains the reason why there is an increase in accuracy when the number of intervals of both equal width and equal frequency increase from 5 to 10. From the findings of bias-variance decomposition in (Flores et al., 2011), we can further deduce the reasoning behind the excellent performance of AODE. It is said that AODE has the lowest error rates.

In the same vein, our empirical experiments involved the investigation of how the size of the training sample size affects the discretization process and ultimately the classification performance of the Bayesian classifiers. Figure 5-3 to Figure 5-7 shows the behaviour of the accuracy value against the variation of training sample sizes for each Bayesian classifier. Generally, it is observed that each classifier reacts uniquely to the discretization approaches. However, overall the MDL method ranked higher across all the classifiers. Noted also was that

on average lower sample sizes lead to optimistic bias in classification performance across the classifiers. Our results are at variance with the fact that supervised discretization methods result in better classifiers when compared to those trained with unsupervised approaches.

5.2.4 Choice of score function

This section will discuss the results of the experiments investigating how the choice of the score function affects Bayesian network classifiers performance. We commence our discussion by analysing the results depicted in Figure 5-8, which shows the classification accuracy values of the TAN, GBN-K2 and GBN-K2 trained with different score functions. From Figure 5-8 it is noted that the choice of score function did not significantly affect the classification accuracy. As published by (Friedman, Geiger & Goldszmidt, 1997), a particular scoring function might learn a high quality network structure but however exhibit mediocre performance on test instances. Figure 5-9 to Figure 5-11 shows the behaviour of the classification accuracy value against the variation of training sample sizes for each Bayesian classifier. It is observed also that the score function is impacted by changes in the size of the training sample. Across all the experiments, bayes score produced classifiers that gave the best results on test data. Literature explicitly alluded the importance of the score-equivalent property. Carvalho (2009); Liu, Malone & Yuan (2012) emphatically state the property as mandatory. All the scoring functions under study are decomposable but bayes and BDeu are not equivalent. Our results are in harmony with the findings of (Yang & Chang, 2002). In their findings they noted that the property is rather useless because the score-equivalent score metrics produced network structures of inferior performance when compared with the non-score-equivalent score metrics.

5.3 Conclusion

This chapter outlined the classification performance of Bayesian network classifiers under individual and collective empirical configurations. Each experiment was described and discussed in detail and thereafter detailed analysis of the outcomes of the experiments was conducted. 10 graphs were produced from the investigations. Observations and conclusions were drawn based on the results of the experiments performed. The closing chapter of this dissertation is for consolidating the findings discovered in this chapter and outlining the future studies that can be explored in this research.

Chapter 6: Conclusion and Future work

Our research has been an empirical investigation to answer the critical questions: how do the model complexity, size of training set, type of discretization approach and the type of score function individual and collectively affect the classification performance of Bayesian networks classifies in distinguishing between normal and anomaly data? We made use of the widely used NSL-KDD dataset for intrusion detection and real DoS network traffic to explore this question in-depth by performing several experiments. This closing chapter aims to show how the findings of the research tally with the research questions outlined in the first chapter. The chapter also provides a comprehensive synopsis of the highlights of the research. Further, we close this chapter by describing areas of possible further exploration for this study.

6.1 Introduction

Bayesian network classifiers are a customised version of Bayesian networks used for classification tasks. They have been used to implement IDS and have proven to be successful and comparative in performance with other data mining algorithms. The constant quest for the data mining or machine learning community is the implementation of optimal performing classifiers. In this study, there are factors we hypothesized to impact the classification performance of Bayesian network classifiers. These are namely structural complexity, training sample size, the type of discretization approach and the lastly the choice of scoring function used. These factors formed the core of the experiments of this research. There exist discrepancies and ambiguous findings about the impact of the factors on classification performance. These have been reviewed accordingly in Section 3.11. This motivated the researcher to conduct empirical investigations of the impact of the aforementioned factors. The domain of application of the experiments is in the detection of DoS attacks in cloud computing.

6.2 Empirical findings vs research questions

1) How do DoS attacks impact the cloud computing environments?

The research introduced the concept of Cloud computing in Chapter 2. Cloud computing is presently the most important internet centric service model. It comes with a wide variety of benefits that are appealing organisation's business model. However, there exist challenges in terms of security that threaten the cloud technology. DoS attacks are the chief security intrusion

that has stalled the mass adoption of the cloud computing environments. DoS attacks affect the availability of cloud resources. Furthermore they degrade the quality of network connectivity of end users. Their ultimate goal is to handicap the consumption of cloud services by users and the availability the services is affected. Consequently, cloud service providers will incur business losses and ultimately lose their clientele.

2) How do Bayesian networks perform classification tasks?

The research firstly discussed the concepts of Bayesian networks in chapter 3. A few definitions were presents to explain the concept. The use of Bayesian networks were justified for classification tasks because of the following advantages:

- Probability theory.
- Graphical structure.
- Ability to fuse expert knowledge and data to construct models.
- Its support for missing data during induction.

Bayesian network are a customized version of Bayesian networks used for classification tasks. The simplest type of Bayesian network classifiers is NB. Due to the unrealistic assumption, the NB tends to perform poorly in certain domains hence it is augmented to TAN, AODE, GBN etc.

3) How does structural complexity of Bayesian classifier affect the classification performance of Bayesian classifiers?

Bayesian network classifiers come in various forms starting from the simple NB up to the complex variants. As stated in literature, the progression in structural complexity is not straightforward and may not be directly proportional to classification performance of Bayesian classifiers. Also in accordance to the no-free-lunch theorem, performance of a classifier is not universal; it varies most notably with dataset types. With reference to Section 3.2.1, it was mentioned that the structural complexity of the Bayesian network classifiers under study increased from NB to GBN. Findings from our empirical study indicate an improved classification performance from NB to AODE but however drop when it comes to GBN. Madden (2009, p. 12) accounts for the drop in performance of GBN buy explaining in terms of computational complexity stating that, "*…GBN classifiers are more expensive to construct than*

TAN classifiers…". Our findings do however confirm that the structural complexity factor does affect the classification performance of Bayesian network classifiers.

4) Does the size of the training dataset impact the classification performance of Bayesian classifiers?

The empirical study pertaining to the relationship between the training sample size and the classification performance of Bayesian network classifiers was discussed in Section 3.2.2. The findings indicate that varying the training sample size used in the induction of the Bayesian networks classifiers has an apparent impact on their classification performance. In light of tediousness of annotating the training data, scholar presented studies that prove that NB may not necessarily require a large dataset for training. Sordo & Zeng (2005) in their study reveal a dwindling performance of NB as training set size increases. This sparked interest in the researcher to investigate the behaviour of the TAN, AODE and GBN when altering the size of the training set. Contrary to the findings by (Sordo & Zeng, 2005), our experiments show that the performance of NB increased with increasing training set size. This was typical for the TAN, AODE and GBN classifiers too. As discussed earlier on, when inducting Bayesian networks from a dataset, the probabilities will be physical not Bayesian. It is to that end that the size of the training set dictates the accuracy of the probability estimates in the resultant Bayesian network structure. Worst classification performance exhibited at small training sample sizes might be due to the distribution of the instances in the dataset not being representative of the problem domain. Furthermore, poor performance can be attributed to the class imbalance phenomena.

5) Does the choice of discretization technique matter and how is the discretization process affected by the size of the training set?

The experimental results regarding the effects of the type of discretization algorithm on the classification performance of Bayesian network classifiers were discussed in Section 5.2.3. It was observed that discretization techniques do impact the classification performance. This investigation was motivated by the fact that data mining suites e.g WEKA and keel come embedded with a number of discretization algorithms hence the researcher was curious on their impact on the classification ability of build Bayesian networks. There exist discrepancies in literature on the supposed impact on classification performance. Other factors of interest was

whether the number of produced intervals and the number of training instances. We also focussed on the impact of supervised and unsupervised discretization algorithms on classification. We noted that supervised learning algorithms produced poor performing models and the findings are at variance with results from other studies. The proportional k-interval discretization algorithm was generally the best performing discretization method. Also the more the interval produced by a discretization the better the classification ability. Larger training sets imply that more information will be contained within the intervals after the discretization process.

6) Does the choice of scoring function result in better classifiers and how are the score metrics affected by the size of the training set?

The empirical results concerning the impact of the score function on the classification performance of Bayesian network classifiers were discussed in section 5.2.4. We investigated and obtained that the score function generally does not impact the classification performance of Bayesian network classifiers. This investigation was motivated by the curiosity invoked by several research papers that are on the development of novel score function or metrics. This sparked interest in whether they in anyhow impact the classification performance of classifiers. It was however discovered that the investigated that the score functions under study did not affect the classification ability of the classes of Bayesian networks in this study.

6.3 Recommendations and Future work

During the course of study any field of interest, various questions will remain unanswered. With respect to this study, our findings have considerable areas that need further exploration. We can extend our study by investigating Bayesian network classifiers as multiclass classifiers. We would like to empirically investigate whether they will provide better classification performance. This study can be modified to study how the Bayesian networks behave when applied to detect other intrusion attacks categories. Furthermore, investigations can be done on the impact of the discretization process during the training phase instead after prior training. The NSL-KDD dataset used as our training set has 41 features. In this study we only managed to derive 5 features. Therefore to improve the quality of our empirical study, given the necessary algorithms for converting packet-level data to connection-level data, we need to embark on 41 features from the live packet capture. The greatest challenge in data mining is in the annotation of the training

data. It is therefore tedious as well as expensive to label large training samples required for optimum classification performance. With this in mind, further exploration can be done to determine how small training samples can be made to augment the classification performance of Bayesian network classifiers.

6.4 Conclusion

Bayesian networks have been applied in a wide-range of applications. The focal point of this research has been to apply them in detecting DoS attacks in cloud computing environments. Literature provides numerous advantages that make Bayesian networks to be the better data mining algorithm. They have proven to be excellent in classification tasks. This research embarked on the analysis of the impact of structural complexity, training sample size, discretization algorithm and score functions on the classification performance in detecting DoS attacks. The hope of the researcher is that this study will provide sound "ecosystem" modelling choices for researchers intending to apply any class of Bayesian network in any field of study. Poor choices of the sample size, discretization technique will have adverse effects on the classification accuracy of chosen classifier. By virtue of our research, the induction of Bayesian networks should be an easy task since the parameters involved have been analysed and presented. For optimal performance of Bayesian network classifiers, it is highly imperative to be acquainted with the options of the involved parameters and the associated issues. Thus, the general rules outlined can be essential for novice machine learners.

References

Abliz, M. 2011. *Internet Denial of Service Attacks and Defense Mechanisms*. Available: http://people.cs.pitt.edu/~mehmud/docs/abliz11-TR-11-178.pdf.

Acid, S., Campos, L.M. De, Ferna, J.M. & Rodrı, S. 2004. A comparison of learning algorithms for Bayesian networks : a case study based on data from an emergency medical service. *Artificial Intelligence in Medicine*. 30:215–232. DOI: 10.1016/j.artmed.2003.11.002.

Aggarwal, C.C. 2014. *Data Classification Algorithms and Applications*. New York, USA: CRC Press.

Alfantookh, A.A. 2006. DoS Attacks Intelligent Detection using Neural Networks. *Journal of King Saud University - Computer and Information Sciences*. 18:31–51. DOI: 10.1016/S1319-1578(06)80002-9.

Alina, M. & Popescu, D.E. 2013. Evaluation of Experiments on Detecting Distributed Denial of Service (DDoS) Attacks in Eucalyptus Private Cloud. *Soft Computing Applicatons*. AISC 195:367–379.

Van Allen, T. & Greiner, R. 2000. Model Selection Criteria for Learning Belief Nets: An Empirical Comparison. In *Proceedings of the Seventeenth International Conference on Machine Learning*. 1047–1054.

Alotaibi, K.H. 2015. Threat in Cloud- Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attack , and Security Measures. *Journal of Emerging Trends in Computing and Information Sciences*. 6(5):241–244.

Arjunwadkar, M. & Kulkarni, R. V. 2010. The Rule Based Intrusion Detection and Prevention Model for Biometric System. *Emerging Trends in Computing and Information Sciences*. 1(2):117–120.

Badger, L., Grance, T., Patt Corner, R. & Voas, J. 2011. Cloud Computing Synopsis and Recommendations. *Recommendations of the National Institute of Standards and Technology*. C O M P:84. DOI: 2012.

Baesens, B., Verstraeten, G., Van den Poel, D., Egmont-Petersen, M., Van Kenhove, P. & Vanthienen, J. 2004. Bayesian network classifiers for identifying the slope of the customer lifecycle of long-life customers. *European Journal of Operational Research*. 156(2):508–523. DOI: 10.1016/S0377-2217(03)00043-2.

Baig, Z.A., Shaheen, A.S. & Abdelaal, R. 2011. One-Dependence Estimators for Accurate Detection of Anomalous Network Traffic. *International Journal for Information Security Research (IJISR)*. 1(4):202–210.

Barbara, D., Wu, N. & Jajodia, S. 2001. Detecting Novel Network Intrusions Using Bayes Estimators. In *First SIAM Conference on Data Mining*. Chicago. 1–17.

Barbará, D. 2001. Special issue on data mining for intrusion detection and threat analysis. *ACM SIGMOD Record*. 30(4):4. DOI: 10.1145/604264.604266.

Beleites, C., Neugebauer, U., Bocklitz, T., Kra, C. & Popp, J. 2013. Sample Size Planning for Classification Models. DOI: DOI: 10.1016/j.aca.2012.11.007.

Bennett, C. 2012. *The Interaction of Entropy-Based Discretization and Sample Size : An Empirical Study*. Available: http://http//arxiv.org/ftp/arxiv/papers/1201/1201.1450.pdf.

Bhattacharyya, D.K. & Kalita, J.K. 2014. *Network Anomaly Detection A Machine Learning Perspective*. CRC Press.

Bielza, C. & Larrañaga, P. 2014. Discrete Bayesian Network Classifiers: A Survey. *ACM Computing Surveys (CSUR)*. 47(1):5. DOI: 10.1145/2576868.

Bishop, C.M. 2006. *Pattern Recognition and Machine Learning*. New York: Springer.

Bogdanoski, M. & Risteski, A. 2011. Wireless Network Behavior under ICMP Ping Flood DoS Attack and Mitigation Techniques. *International Journal of Communication Networks and Information Security (IJCNIS)*. 3(1):17–24.

Boland, D.J. 2007. Data Discretization Simplified : Randomized Binary Search Trees for Data Preprocessing. West Virginia University. Available: http://menzies.us/pdf/BolandThesis.pdf.

Bouckaert, R. 1994. *Probabilistic Network Construction Using the MInimum Description Length Principle*.

Bouckaert, R.R. 2008. Bayesian network classifiers in Weka for version 3-5-7. *Artificial Intelligence Tools*. Available: http://www.cs.waikato.ac.nz/~remco/weka.bn.pdf.

Boullé, M. 2005. A Bayes optimal approach for partitioning the values of categorical attributes. *Journal of machine learning research*. 6(2):1431–1452. Available: http://www.jmlr.org/papers/volume6/boulle05a/boulle05a.pdf.

Bozdogan, H. 1987. Model Selection and Akaike's Information criterion (AIC): The general theory and its analytical extensions. *Psychometrika*. 52(3):345–370.

Brain, D. & Webb, G.I. 1999. On The Effect of Data Set Size on Bias And Variance in Classification Learning. *Proceedings of the Fourth Australian Knowledge Acquisition Workshop (AKAW '99)*. 117–128.

Brenner, E. & Sontag, D. 2013. SparsityBoost: A New Scoring Function for Learning Bayesian Network Structure. *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*. Available: http://www.cs.nyu.edu/~dsontag/papers/BrennerSontag_uai13.pdf.

Bringas, P.G. & Santos, I. 2010. Bayesian Networks for Network Intrusion Detection. 229–245. Available: http://cdn.intechopen.com/pdfs-wm/11952.pdf.

Bringas, P.G., Penya, Y.K., Paraboschi, S. & Salvaneschi, P. 2008. Bayesian Networks-Based Misuse and Anomaly Prevention System. In *Proceedings of the Tenth International Conference on Enterprise Information Systems*. Barcelona, Spain. 62–69.

Brugger, S. & Chow, J. 2007. An assessment of the DARPA IDS Evaluation Dataset using Snort. *UCDAVIS department of Computer Science*. 1–19. DOI: 10.1.1.94.674.

Buntine, W. 1996. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*. 8(2):195–210. DOI: 10.1109/69.494161.

Butun, I., Morgera, S. & Sankar, R. 2013. A Survey of Intrusion Detection Systems in Wireless Sensor Networks. *Communications Surveys Tutorials, IEEE*. PP(99):1–17. DOI: 10.1109/SURV.2013.050113.00191.

Campos, L.M. De. 2006. A Scoring Function for Learning Bayesian Networks based on Mutual Information and Conditional Independence Tests. *Journal of Machine Learning Research*. 7:2149–2187.

Cano, A., Nguyen, D.T., Ventura, S. & Cios, K.J. 2014. ur-CAIM : improved CAIM discretization for unbalanced and balanced data. *Methodologies and Application*. DOI: 10.1007/s00500-014-1488-1.

Carvalho, A.M. 2009. *Scoring functions for learning Bayesian networks*. Lisbon, Portugal.

Castillo, G. & Gama, J. 2005. Bias Management of Bayesian Network Classifiers. In *Proceedings of 8th International Conference*. D. Science, Ed. Springer Verlag. 70–83.

Cheng, J. & Greiner, R. 1999. Comparing Bayesian network classifiers. In *Proceeding of the 15th International Conference on Uncertainty in Artificial Intelliegence*. Morgan Kaufmann. 101–108.

Chickering, D.M. 1996. Learning Bayesian Networks is NP-Complete. In *Learning from Data: Artificial Intelligence and Statistics V*. D. Fisher & H. Lenz, Eds. Springer-Verlag. 121–130.

Chickering, D.M., Heckerman, D. & Meek, C. 2013. A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings of the Thirteenth conference on Uncertainty in Artificial Intelligence*. Providence, Rhode Island. 80–89.

Citrix. 2014. *Citrix NetScaler : A Powerful Defense Against Denial of Service Attacks*. Available: https://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/citrix-netscaler-a-powerful-defense-against-denial-of-service-attacks.pdf.

Clarke, E.J. & Barton, B.A. 2000. Entropy and MDL Discretization of Continuous Variables for Bayesian Belief Networks. *International Journal of Intelligent Systems*. 15:61–92.

Cooper, G.F. & Herskovits, E. 1992a. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*. 9(4):309–347. DOI: 10.1007/BF00994110.

Cooper, G.F. & Herskovits, E. 1992b. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*. 9:309–347.

CSA. 2010. Cloud security alliance Top Threats to cloud computing. (March):1–14. Available: https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf.

Dally, R., Shen, Q. & Aitken, S. 2011. Learning Bayesian networks : approaches and issues. *The Knowledge Engineering Review*. 26(2):99–157. DOI: 10.1017/S0269888910000251.

Demˇsar, J. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*. 7:1–30.

Deshmukh, R. V. & Devadkar, K.K. 2015. Understanding DDoS Attack & its Effect in Cloud Environment. *Procedia Computer Science*. 49:202–210. DOI: 10.1016/j.procs.2015.04.245.

Divakaran, D.M., Murthy, H.A. & Gonsalves, T.A. 2006. DETECTION OF SYN FLOODING ATTACKS USING LINEAR PREDICTION ANALYSIS. In *Proceedings of the 14th IEEE International Conference on Networks*. 1–6.

Divina, F. & Marchiori, E. 2005. Handling continuous attributes in an evolutionary inductive learner. *IEEE Transactions on Evolutionary Computation*. 9(1):31–43. DOI: 10.1109/TEVC.2004.837752.

Domingos, P. & Pazzani, M. 1997. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*. 29:103–130. DOI: 10.1023/A:1007413511361.

Dougherty, J., Kohavi, R. & Sahami, M. 1995. Supervised and unsupervised discretization of continuous features. *Machine Learning: Proceedings of the Twelfth International Conference*. 54(2):194–202. DOI: citeulike-article-id:1176665.

Douligeris, C. & Mitrokotsa, A. 2004. DDoS attacks and defense mechanisms: Classification and state-of-the-art. *Computer Networks*. 44(5):643–666. DOI: 10.1016/j.comnet.2003.10.003.

Duda, R.O., Hart, P.E. & Stork, D.G. 1997. *Pattern classification*. 2nd ed. New York: Wiley.

Eskridge, T., Lecoutre, D., Johnson, M. & Bradshaw, J.M. 2009. Network Situational Awareness : A Representative Study. In *Human-computer interaction and visualization (HCIV 2009)*. Kaiserslautern, Germany.

Fayyad, U.M. & Irani, K.B. 1993. Multi-Interval Discretization of Continuos-Valued Attributes for Classification Learning. In *Proceedings of the International Joint Conference on Uncertainty in AI*. San Francisco, CA: Morgan Kaufmann. 1022–1027.

Fernandes, D.A.B., Soares, L.F.B., Gomes, J. V, Freire, M.M. & Inacio, P.R.M. 2014. Security Issues in Cloud Environments — A Survey. *International Journal of Information Security (IJIS)*. 113–179. DOI: 10.1007/s10207-013-208-7.

Figueroa, R.L. & Zeng-treitler, Q. 2012. Predicting sample size required for classification performance. *BMC Medical Informatics and Decision Making*. 12(1):8. DOI: 10.1186/1472-6947-12-8.

Flores, M.J., Gámez, J.A., Martínez, A.M. & Puerta, J.M. 2011. Handling numeric attributes when comparing Bayesian network classifiers : does the discretization method matter ? *Applied Intelligence*. 372–385. DOI: 10.1007/s10489-011-0286-z.

Foody, G.M., Mathur, A., Sanchez-Hernandez, C. & Boyd, D.S. 2006. Training set size requirements for the classification of a specific class. *Remote Sensing of Environment*. 104(1):1–14. DOI: 10.1016/j.rse.2006.03.004.

Friedman, N. & Goldszmidt, M. 1996. Building Classifiers using Bayesian Networks. In *Proceedings of the National Conference on Artificial Intelligence*. 1277–1284.

Friedman, N., Geiger, D. & Goldszmidt, M. 1997. Bayesian Network Classifiers. *Machine learning*. 29:131–163. DOI: 10.1023/A:1007465528199.

García, S., Luengo, J., Sáez, J.A., López, V. & Herrera, F. 2013. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*. 25(4):734–750. DOI: 10.1109/TKDE.2012.35.

Gonzalez, N., Miers, C., Redigolo, F., Carvalho, T., Simplicio, M., Naslund, M. & Pourzandi, M. 2011. A Quantitative Analysis of Current Security Concerns and Solutions for Cloud Computing. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science*. 231–238. DOI: 10.1109/CloudCom.2011.39.

Gruschka, N. & Jensen, M. 2010. Attack surfaces: A taxonomy for attacks on cloud services. *Proceedings - 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD 2010*. 276–279. DOI: 10.1109/CLOUD.2010.23.

Hacibeyoglu, M., Arslan, A. & Kahramanli, S. 2011. Improving Classification Accuracy with Discretization on Datasets Including Continuous Valued Features. *World Academy of Science, Engineering and Technology 78 2011*. 5(6):555–558.

Haghanikhameneh, F., Hassany, P.H.S., Khanahmadliravi, N. & Mousavi, S.A. 2012. A Comparison Study between Data Mining Algorithms over Classification Techniques in Squid Dataset. *International Journal of Artificial Intelligence (IJAI)*. 9.

Hall, M., National, H., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. & Witten, I.H. 2009. The WEKA Data Mining Software : An Update. *SIGKDD Explorations*. 11(1):10–18. DOI: 10.1145/1656274.1656278.

Hastie, T., Tibshirani, R. & Friedman, J. 2008. *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. 2nd Editio ed. V. 1. Springer, Ed. DOI: 10.1007/b94608.

Hastie, T., Tibshirani, R. & Friedman, J. 2009. *The Elements of Statistical Learning*. 2nd ed. Springer Verlag.

Heckerman, D. 1996. A Tutorial on Learning With Bayesian Networks. *Innovations in Bayesian Networks*. 1995(November):33–82. DOI: 10.1007/978-3-540-85066-3.

Heckerman, D. 1997. Bayesian Networks for Data Mining. *Data Mining and Knowledge Discovery*. 1(1):79–119.

Heckerman, D., Geiger, D. & Chickering, D.M. 1994. Learning Bayesian Networks : The Combination of Knowledge and Statistical Data. In *In Uncertainty in Artificial Intelligence, Proceedings of the Tenth Conference*. San Francisco, CA: Morgan Kaufman. 293–301.

Hogan, M., Liu, F., Sokol, A. & Tong, J. 2011. *NIST Cloud Computing Standards Roadmap*. Available: http://www.nist.gov/itl/cloud/upload/NIST_SP-500-291_Jul5A.pdf.

Hoyt, P.J. 2008. Discretization and Learning of Bayesian Networks unsing Stochastic Search, with Application to Base Realignment and Closure (BRAC). George Mason University. Available: http://hdl.handle.net/1920/3141.

Ismail, M.N., Aborujilah, A., Musa, S. & Shahzad, A. 2013. Detecting Flooding based DoS Attack in Cloud Computing Environment using Covariance Matrix Approach. In *Proceedings of the ACM 7th International Conference on Ubiquitous Information Management and Communication*. 4–9.

Isnin, F.I. 2011. A study on wireless communication error performance and path loss prediction. University of Plymouth. Available: https://pearl.plymouth.ac.uk/handle/10026.1/324.

Kaya, E., Findik, O., Babaoğlu, I. & Arslan, A. 2011. Effect of discretization method on the diagnosis of Parkinson's disease. *International Journal of Innovative Computing, Information and Control*. 7(8):4669–4678.

Kayacik, H., Zincir-Heywood, a N. & Heywood, M.I. 2005. Selecting Features for Intrusion Detection : A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets. *Proceedings*

of the Third Annual Conference on Privacy Security and Trust PST2005. 3–8. DOI: 10.1.1.66.7574.

KDD'99 Dataset. n.d. Available: http://kdd.ics.uci.edu/databases/.

Keogh, E.J. & Pazzani, M.J. 1999. Learning Augmented Bayesian Classifiers : A Comparison of Distribution-based and Classification-based Approaches. In *Proceedining of the 7th International Workshop on AI and Statistics*. 225–230.

Kerber, R. 1992. Chimerge: Discretization of numeric attributes. *Proceedings of the tenth national conference on Artificial intelligence*. 123–128. Available: http://dl.acm.org/citation.cfm?id=1867154\npapers2://publication/uuid/F1C11C06-12F0-4A83-82D3-CC62A52FBE75.

Khalil, I., Khreishah, A. & Azeem, M. 2014. Cloud Computing Security: A Survey. *Computers*. 3(1):1–35. DOI: 10.3390/computers3010001.

Khan, L., Awad, M. & Thuraisingham, B. 2007. A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB Journal*. 16(4):507–521. DOI: 10.1007/s00778-006-0002-5.

Khorshed, T.M., Ali, S.A.B.M. & Wasimi, S.A. 2012. Classifying different denial-of-service attacks in cloud computing using rule-based learning. *Security and Communications Networks*. 2(September):71–81. DOI: 10.1002/sec.

Kim, W. & Korea, S. 2009. Cloud Computing : Today and Tomorrow. *Journal of Object Technology*. 8(1):65–72. DOI: 10.5381/jot.2009.8.1.c4.

Koc, L. & Carswell, A.D. 2015. Application of an AODE Based Classifier to Detect DOS Attacks. *International Journal of Computer Science and Network Security (IJCSNS)*. 15(2):24–28.

Koller, D. & Friedman, N. 2009. *Probabilistic Graphical Models Principles and Techniques*. Massachusetts Institute of Technology.

Korb, K.B. & Nicholson, A.E. 2004. *Bayesian Artificial Intelligence*. London: Chapman & Hall/CRC.

Kothari, C.R. 1990. *Research Methodology Methods and Techniques*. 2nd ed. New Delhi, India: New Age International.

Kotsiantis, S.B., Kanellopoulos, D. & Pintelas, P.E. 2006. Data preprocessing for Supervised Leaning. *International Journal of Computer Science*. 1(2):1–7. DOI: 10.1080/02331931003692557.

Kurgan, L. & Cios, K.J. 2001. Discretization algorithm that uses class-attribute interdependence maximization. *Proc. of the 2001 International Conference on Artificial Intelligence (ICAI-2001)*. 980–987. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.4872.

Lam, W. & Bacchus, F. 1994. Learning Bayesian Belief Networks: An Approach based on the MDL Principle. *Computation Intelligence Vol 10:4, pages 269--293*. 10:0–31.

Langley, P., Iba, W. & Thompson, K. 1992. An analysis of Bayesian classifiers. In *Proceedings of the National Conference on Artificial Intelligence*. A. Press, Ed. San Jose. 223–223. DOI: citeulike-article-id:932362.

Lee, C.H. 2007. A Hellinger-based discretization method for numeric attributes in classification learning. *Knowledge-Based Systems*. 20(4):419–425. DOI: 10.1016/j.knosys.2006.06.005.

Lee, W., Stolfo, S.J. & Mok, K.W. 1999. Mining in a data-flow environment: Experience in network intrusion detection. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. (212):114–124. DOI: http://doi.acm.org/10.1145/312129.312212.

Lee, W., Stolfo, S.J., Mok, K.W., Street, W. & York, N. 1999. A Data Mining Framework for Building Intrusion Detection Models ∗. In *In Proceedings of the 1999 IEEE Symposium on Security and Privacy*. Available: http://ids.cs.columbia.edu/sites/default/files/wenke-ieee99.pdf.

Lerner, B. & Malka, R. 2011. INVESTIGATION OF THE K2 ALGORITHM IN LEARNING BAYESIAN NETWORK CLASSIFIERS. *Applied Artificial Intelligence*. 1(25):74–96. DOI: 10.1080/08839514.2011.529265.

Lippmann, R. & Haines, J. 2000. Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation. *Recent Advances in Intrusion Detection, Third International Workshop*. 16–182. DOI: 10.1007/3-540-39945-3_11.

Lippmann, R., Haines, J.W., Fried, D.J., Korba, J. & Das, K. 2000. The 1999 DARPA o € -line intrusion detection evaluation. 34:1–22. Available: https://www.ll.mit.edu/ideval/files/1999Eval-ComputerNetworks2000.pdf.

Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., et al. 2000. Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*. 2. DOI: 10.1109/DISCEX.2000.821506.

Liu, H., Hussain, F., Tan, C.L.I.M. & Dash, M. 2002. Discretization : An Enabling Technique. *Data Mining and Knowledge Discovery*. 6:393–423.

Liu, Z., Malone, B. & Yuan, C. 2012. Empirical evaluation of scoring functions for Bayesian network model selection. *BMC Bioinformatics*. 13(Suppl 15):S14. DOI: 10.1186/1471-2105-13-S15-S14.

Lonea, A.M., Popescu, D.E. & Tianfield, H. 2013. Detecting DDoS Attacks in Cloud Computing Environment Dempster-Shafer Theory (DST). *International Journal of Computing and communication*. 8(1):70–78.

Love, B.C. 2002. Comparing supervised and unsupervised category learning. *Psychonomic bulletin & review*. 9(4):829–835. DOI: 10.3758/BF03196342.

Madden, M.G. 2003. The Performance of Bayesian Network Classifiers Constructed Using Different Techniques. *In Working notes of the ECML/PKDD-03 workshop on*. 59–70.

Madden, M.G. 2009. Knowledge-Based Systems On the classification performance of TAN and general Bayesian networks. *Knowledge-Based Systems*. 22(7):489–495. DOI: 10.1016/j.knosys.2008.10.006.

Mahoney, M. V & Chan, P.K. 2003. An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. *In Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection*. 2820(Ll):220–237. DOI: 10.1007/b13476.

Maletic, J.I. & Marcus, A. 2005. Data Cleansing - A prelude to knowledge discovery. In *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*. O. Maimon & L. Rokach, Eds. Kluwer Academic Publishers. 21–36. DOI: 10.1007/0-387-25465-x_2.

Malone, B., Jarvisalo, M. & Myllymaki, P. 2015. Impact of Learning Strategies on the Quality of Bayesian Networks : An Empirical Evaluation. In *In Proceedings of the 31st Conference on UNcertainity in Artificial Intelligence (UAI 2015)*.

Marcot, B.G. 2012. Metrics for evaluating performance and uncertainty of Bayesian network models. *Ecological Modelling*. 230:50–62.

Melton, J., Buxton, S., Samet, H., Teorey, T.J., Lightstone, S.S., Nadeau, T.P., Celko, J., Ralf, G., et al. 1999. *Data Mining : Concepts and Techniques*. 2nd Editio ed. Morgan Kaufmaan Publishers.

Miltov, Ii., Ivanova, K., Markov, K., Velychko, V., Stanchev, P. & Vanhoof, K. 2009. COMPARISON OF DISCRETIZATION METHODS FOR PREPROCESSING DATA FOR PYRAMIDAL GROWING NETWORK CLASSIFICATION METHOD Ilia Mitov , Krassimira Ivanova , Krassimir Markov ,. *New Trends in Intelligent Technology*. (14):31–39. Available: http://foibg.com/ibs_isc/ibs-14/ibs-14-p04.pdf.

Mirkovic, J. 2003. D-WARD: source-end defense against distributed denial-of-service attacks. University of California. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.7002&rep=rep1&type=pdf .

Mitchel, T.M. 1997. *Machine Learning*. McGraw-Hill. DOI: 10.1007/978-3-642-21004-4_10.

Mizianty, M.J. & Kurgan, L.A. 2010. Discretization as the enabling technique for the ¨ ve Bayes and semi-Naı Naı classification. 25:421–449. DOI: 10.1017/S0269888910000329.

Mizianty, M., Kurgan, L. & Ogiela, M. 2008. Comparative analysis of the impact of discretization on the classification with aive Bayes and semi- aïve Bayes classifiers. In *In Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications*. 823–828.

Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A. & Rajarajan, M. 2013. A survey of intrusion detection techniques in Cloud. *Journal of Network and Computer Applications*. 36(1):42–57. DOI: 10.1016/j.jnca.2012.05.003.

Moore, A.W. & Zuev, D. 2005. Internet Traffic Classification Using Bayesian Analysis Techniques. In *In SIGMETRIC'05*. Banff, Canada.

Moyo, V. & Sibanda, K. 2015. Training Set Size for Generalization Ability of Artificial Neural Networks in Forecasting TCP / IP Traffic Trends. *International Journal of Computer Applications*. 113(13):14–19.

Murphy, K.P. 2001. The bayes Net Toolkit for Matlab. *In Computing science and statistics*. 33.

Navaz, A.S.S., Sangeetha, V. & Prabhadevi, C. 2013. Entropy based Anomaly Detection System to Prevent DDoS Attacks in Cloud. *International Journal of Computer Applications*. 62(15):42–47.

Ostermann, S. 2003. Available: http://www.tcptrace.org.

OwnCloud. 2015. Available: https://owncloud.com.

Palmer, J. 2011. Using Live Packet Capture. In *Data mining in Bioinfomatics*. 1–4. Available: http://www.e-ijaet.org/media/33I20-IJAET0520947_v7_iss2_568-574.pdf.

Panda, M. & Patra, M.R. 2007. Network Intrusion Detection Using Naïve Bayes. *Journal of Computer Science*. 7(12):258–263.

Paulson, P., Carroll, T.E., Sivaraman, C., Neorr, P., Unwin, S.D. & Hossain, S. 2011. Simplifying probability elicitation and uncertainty modeling in bayesian networks. *CEUR Workshop Proceedings*. 710:114–119.

Paxson, V. 1999. Bro: a system for detecting network intruders in real-time. *Computer Networks*. 31(23-24):2435–2463. DOI: 10.1016/S1389-1286(99)00112-7.

Pearce, M., Zeadally, S. & Hunt, R. 2013. Virtualization: Issues, Security Threats, and Solutions. *ACM Computing Surveys*. 45(2):1–39. DOI: 10.1145/2431211.2431216.

Pearl, J. 1993. B e l i e f n e t w o r k s revisited *. *Artificial Intelligence*. 59:49–56.

Peddabachigari, S., Abraham, A. & Thomas, J. 2004. Intrusion detection systems using decision trees and support vector machines. *International Journal of Applied …*. 1–16. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.4079&rep=rep1&type=pdf.

Peña, J.M., Björkegren, J. & Tegnér, J. 2005. Learning dynamic Bayesian network models via cross-validation. *Pattern Recognition Letters*. 26(14):2295–2308. DOI: 10.1016/j.patrec.2005.04.005.

Pyle, D. & Cerra, D.D. 1999. *Data Preparation for Data Mining*. Morgan Kaufmaan Publishers.

Qaisar, S. & Khawaja, K.F. 2012. Cloud Computing - Network/Security Threats and Countermeasures. *Interdisciplinary Journal of Contemporary Research in Business*. 3(9):1323–1329.

Ramgovind, S., Eloff, M. & Smith, E. 2010. The management of security in cloud computing. *Information Security for South Africa (ISSA)*. 1–7. DOI: 10.1109/ISSA.2010.5588290.

Reddy, E.K. & Iaeng, M. 2013. Neural Networks for Intrusion Detection and Its Applications. In *World Congress on Engineering*. V. II. London, UK.

Riesen, M. & Serpen, G. 2009. A Bayesian Belief Network Classifier for Predicting Victimization in National Crime Victimization Survey. In *In Proceedings of the 2009 International Conference on Artificial Intelligence*. Las Vegas, New York. 648–652.

Rodriguez, J.D. 2013. Advances in Error Estimation and Multi-Dimensional Supervised Classification. University of Basque Country. Available: www.sc.ehu.es/ccwbayes/.../2013_phd_dissertation_jd_rodriguez.pdf.

Rossi, R.A. 2013. *Simple Bayesian Network Classifiers*. Available: http://www.ryanrossi.com/random/data-mining/simple-bayesian-network-classifiers.pdf.

Roure, J. 2004. Incremental Methods for Bayesian Network Structure Learning. Universitat Politecnica de Catalunya.

Ryan, J., Lin, M. & Mikkulainen, R. 1998. Intrusion Detection with Neural Networks. *Advances in Neural Information Processing Systems*. 942–949. DOI: citeulike-article-id:9827152.

Sabhnani, M. & Serpen, G. 2003. Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context. *Proceedings of International Conference on Machine Learning: Models, Technologies, and Applications (MLMTA)*. 209–215. DOI: citeulike-article-id:9827151.

Sanfilippo, S. 1998. Available: www.hping.org.

Sarkar, S., Srivastav, a & Shashanka, M. 2013. Maximally Bijective Discretization for data-driven modeling of complex systems. *Proceedings of the American Control Conference*. 2674–2679. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84883540502&partnerID=40&md5=63107cf9f4a7e8ebb8686fd56d93b056.

Sathya, R. & Abraham, A. 2013. Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. *International Journal of advanced Research in Artificial Intelligence*. 2(2):34–38. DOI: 10.14569/IJARAI.2013.020206.

Scarfone, K. & Mell, P. 2007. Guide to Intrusion Detection and Prevention Systems (IDPS) Recommendations of the National Institute of Standards and Technology. *Nist Special Publication*. 800:94. Available: http://www.reference.com/go/http://csrc.ncsl.nist.gov/publications/nistpubs/800-94/SP800-94.pdf.

Scott, S.L. 2004. A Bayesian paradigm for designing intrusion detection systems. 45:69–83. DOI: 10.1016/S0167-9473(03)00177-4.

Selman, B. & Gomes, C.P. 2002. Hill-climbing Search. In *Nature Encyclopedia of Cognition*. 333–336.

Sen, J. 2013. Security and Privacy Issues in Cloud Computing. *Architectures and Protocols for Secure Information Technology*. (iv):42.

Shahzad, F. 2014. State-of-the-art Survey on Cloud Computing Security Challenges, Approaches and Solutions. *Procedia Computer Science*. 37:357–362. DOI: 10.1016/j.procs.2014.08.053.

Shaughnessy, P. & Livingston, G. 2005. *Evaluating the Causal Explanatory Value of Bayesian Network Structure Learning Algorithms University of Massachusetts , Lowell University of Massachusetts , Lowell*. Available: http://www.aaai.org/Papers/JAIR/Vol35/JAIR-3509.pdf.

Shea, R. 2012. Understanding the Impact of Denial of Service Attacks on Virtual Machines.

Sordo, M. & Zeng, Q. 2005. On Sample Size and Classification Accuracy: A Performance Comparison. *Biological and Medical Data Analysis*. 3745:193–201. DOI: 10.1007/11573067_20.

Specht, S.M. & Lee, R.B. 2004. Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures. In *Proceedings of the 17th International Conference of Parallel and Distributed Computing Systems*. 543–550.

Stefanini, F.M. 2008. Eliciting expert beliefs on the structure of a Bayesian network. In *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models*. Available: http://parallel.vub.ac.be/bnatwork/BNatWork/Publications_files/pgm08_proceedings.pdf#page=285.

Subashini, S. & Kavitha, V. 2011. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*. 34(1):1–11. DOI: 10.1016/j.jnca.2010.07.006.

Suzuki, J. 1993. A construction of Bayesian networks from databases based on an MDL principle. *Proceedings of the Ninth international conference on*. 266–273. Available: http://dl.acm.org/citation.cfm?id=2074506.

Tavallaee, M., Bagheri, E., Lu, W. & Ghorbani, A.A. 2009. A Detailed Analysis of the KDD CUP 99 Data Set. (Cisda):1–6.

Tillander, A. & Pavlenko, T. 2012. Effect of data discretization on the classification accuracy in a high-dimensional framework References. *International Journal of INtelligent Systems*. 27(4):355–374.

Uusitalo, L. 2006. Advantages and challenges of Bayesian networks in environmental modelling. *Ecological Modelling*. 3:312–318. DOI: 10.1016/j.ecolmodel.2006.11.033.

Vaquero, L.M., Rodero-Merino, L., Caceres, J. & Lindner, M. 2008. A break in the clouds. *ACM SIGCOMM Computer Communication Review*. 39(1):50. DOI: 10.1145/1496091.1496100.

Vaquero, L.M., Rodero-merino, L., Caceres, J. & Lindner, M. 2009. A Break in the Clouds : Towards a Cloud Definition. *ACM SIGCOMM Computer Communication Review*. 39(1):50–55.

Vidhya, V. 2014. A Review of DOS Attacks in Cloud Computing. *IOSR Journal of Computer Engineering (IOSR-JCE)*. 16(5):32–35.

Webb, G.I., Boughton, J.R. & Wang, Z. 2005. Not so naive Bayes: Aggregating one-dependence estimators. *Machine Learning*. 58(1):5–24. DOI: 10.1007/s10994-005-4258-6.

Wolpert, D.H. & Macready, W.G. 1997. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*. 1(1):67–82. Available: http://ti.arc.nasa.gov/m/profile/dhw/papers/78.pdf.

Wu, X. 1996. A Bayesian Discretizer for Real-Valued Attributes. *The Computer Journal*. 39(8).

Yang, S. & Chang, K.C. 2002. Comparison of score metrics for Bayesian network learning. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.* 32(3):419–427. DOI: 10.1109/TSMCA.2002.803772.

Yang, Y. & Webb, G.I. 2001. Proportional k-Interval Discretization for. In *In Proceedings of the 12th European Conference on Machine Learning*. Berlin: Springer. 564–575.

Yang, Y. & Webb, G.I. 2002. A Comparative Study of Discretization Methods for Naive-Bayes Classifiers. *Knowledge Acquisition*. 2002:159–173. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.298.

Yu, S. 2013. *Distributed Denial of Service Attack and Defence –*. Springer. DOI: 10.1007/978-1-4614-9491-1_5.

Zhang, Y., Juels, A., Reiter, M. & Ristenpart, T. 2012. Cross-VM side channels and their use to extract private keys. *Proceedings of the 2012 ACM conference on Computer and communication security*. 305–316. DOI: 10.1145/2382196.2382230.

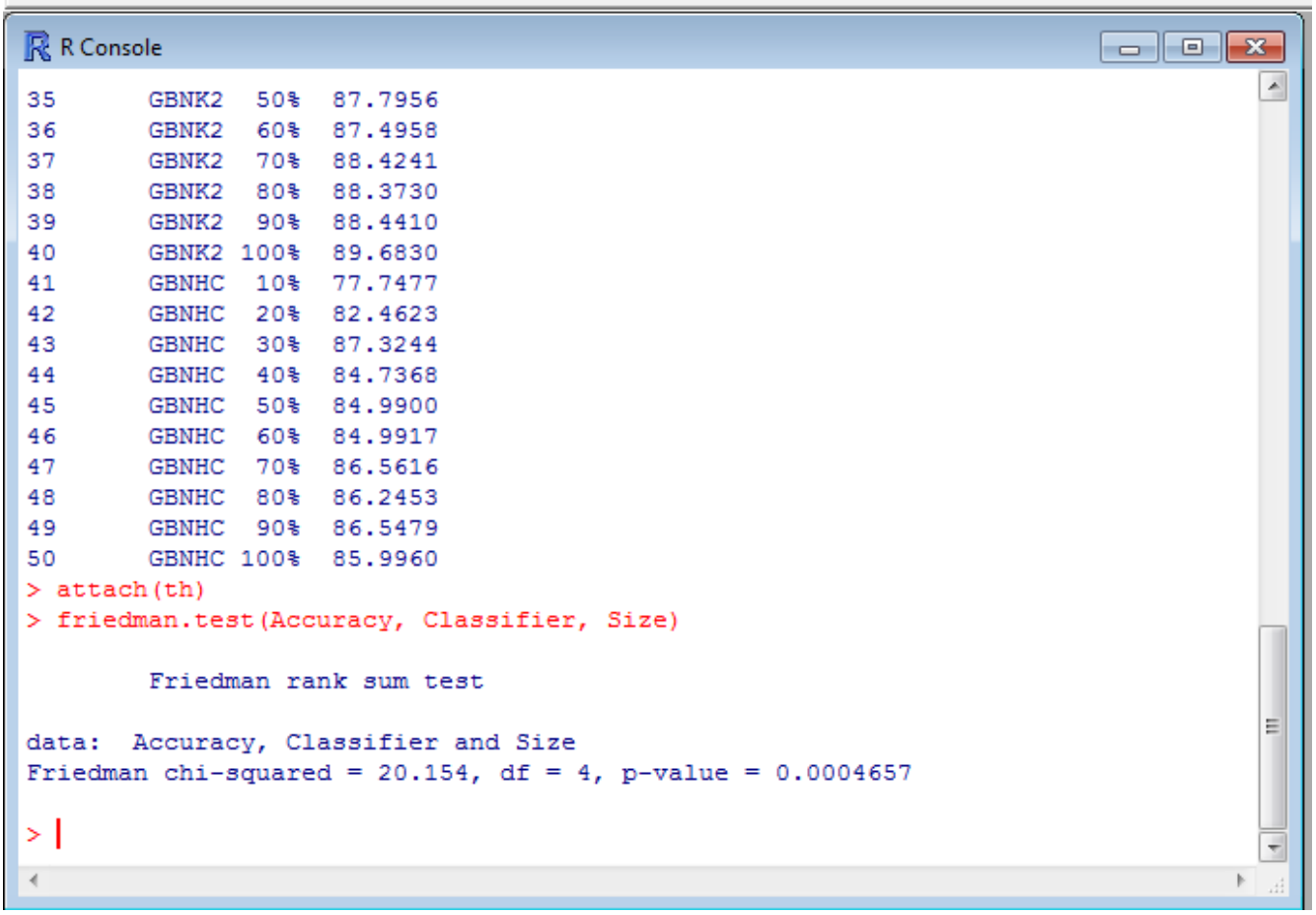
Zheng, Z. & Webb, G.I. 2000. Lazy Learning of Bayesian Rules. *Machine Learning*. 87:53–87.

Zissis, D. & Lekkas, D. 2012a. Addressing cloud computing security issues. *Future Generation Computer Systems*. 28(3):583–592. DOI: 10.1016/j.future.2010.12.006.

Zissis, D. & Lekkas, D. 2012b. Addressing cloud computing security issues. *Future Generation Computer Systems*. 28(3):583–592. DOI: 10.1016/j.future.2010.12.006.

Zunnurhain, K. & Vrbsky, S. 2011. Security in cloud computing. *International Conference in Security and Management*. DOI: 10.1.1.217.9966.

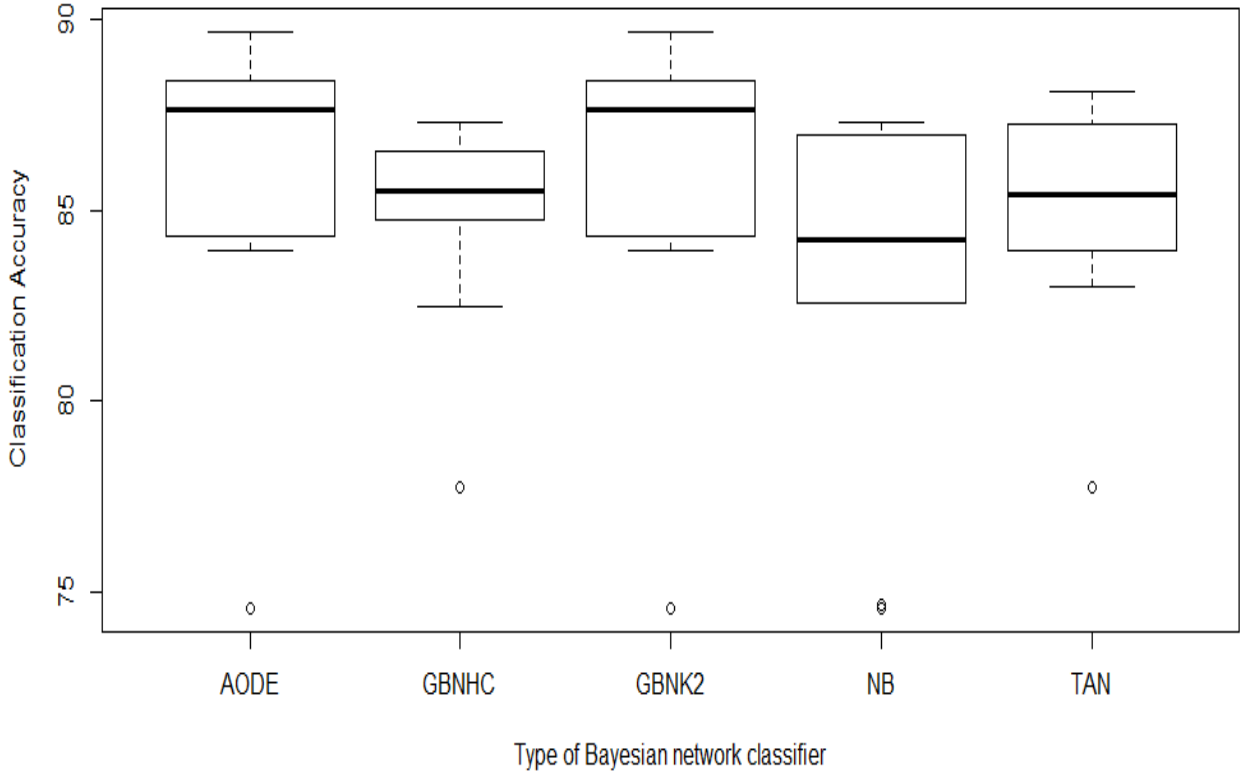
Appendix A



Appendix B

```
> posthoc.friedman.nemenyi.test(Accuracy, Classifier, Size)

        Pairwise comparisons using Nemenyi post-hoc test with q approximation for unreplicated blocked data

data:  Accuracy , Classifier and Size

      AODE   GBNHC  GBNK2  NB
GBNHC 0.0674 -      -      -
GBNK2 1.0000 0.0674 -      -
NB    0.0079 0.9550 0.0079 -
TAN   0.7899 0.5723 0.7899 0.1826

P value adjustment method: none
>
```

Appendix C



Appendix D

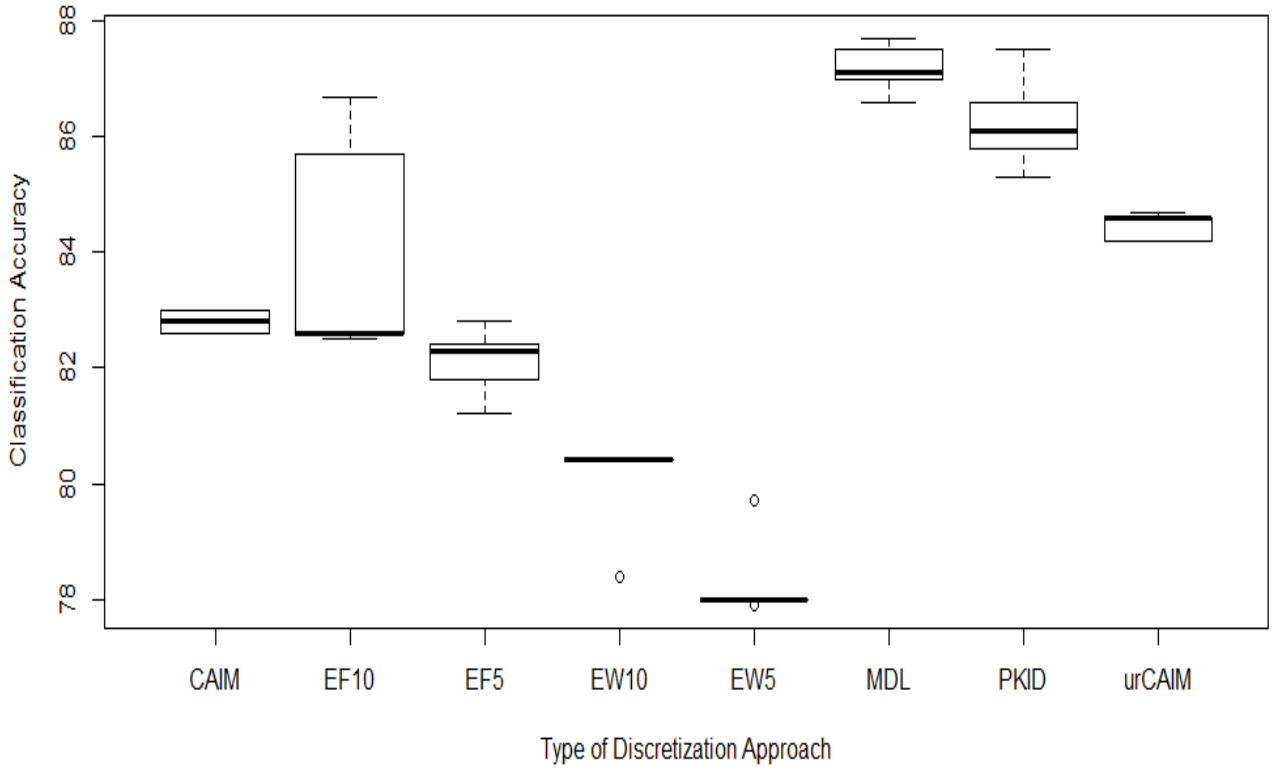
R Console

```
33       AODE           CAIM     82.8
34      GBNK2           CAIM     83.0
35      GBNHC           CAIM     83.0
36         NB         urCAIM     84.6
37        TAN         urCAIM     84.7
38       AODE         urCAIM     84.2
39      GBNK2         urCAIM     84.6
40      GBNHC         urCAIM     84.2
> attach(th)
The following objects are masked from th (pos = 3):

    Accuracy, Classifier, Discretization

The following objects are masked from th (pos = 4):

    Accuracy, Classifier

> friedman.test(Accuracy, Discretization, Classifier)

        Friedman rank sum test

data:  Accuracy, Discretization and Classifier
Friedman chi-squared = 33.56, df = 7, p-value = 2.082e-05

>
```

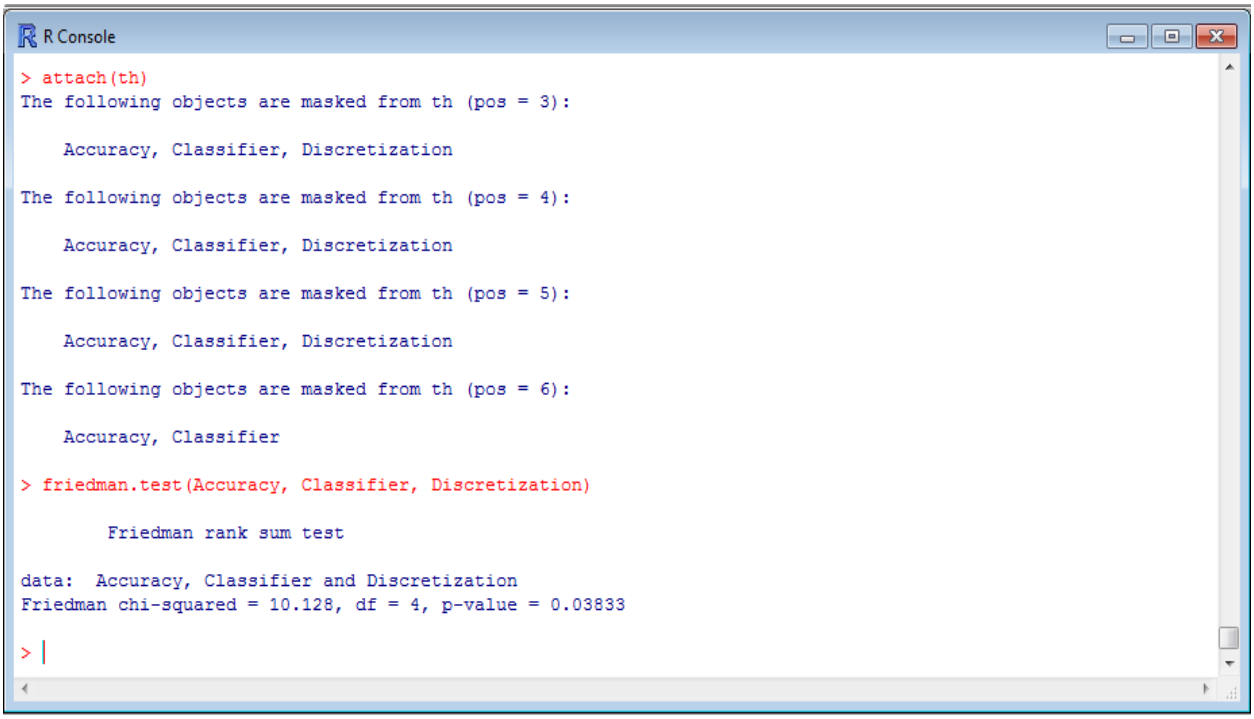
Appendix E

```
> require(PMCMR)
> posthoc.friedman.nemenyi.test(Accuracy, Discretization, Classifier)

        Pairwise comparisons using Nemenyi post-hoc test with q approximation for unreplicated blocked data

data:  Accuracy , Discretization and Classifier

       CAIM    EF10    EF5     EW10    EW5     MDL     PKID
EF10   1.00000 -       -       -       -       -       -
EF5    0.98581 0.95758 -       -       -       -       -
EW10   0.74218 0.61502 0.99674 -       -       -       -
EW5    0.31644 0.21605 0.87718 0.99822 -       -       -
MDL    0.35481 0.48105 0.04090 0.00350 0.00023 -       -
PKID   0.70149 0.81597 0.16237 0.02225 0.00210 0.99958 -
urCAIM 0.99674 0.99958 0.74218 0.28041 0.05974 0.81597 0.97883

P value adjustment method: none
>
```

Appendix F



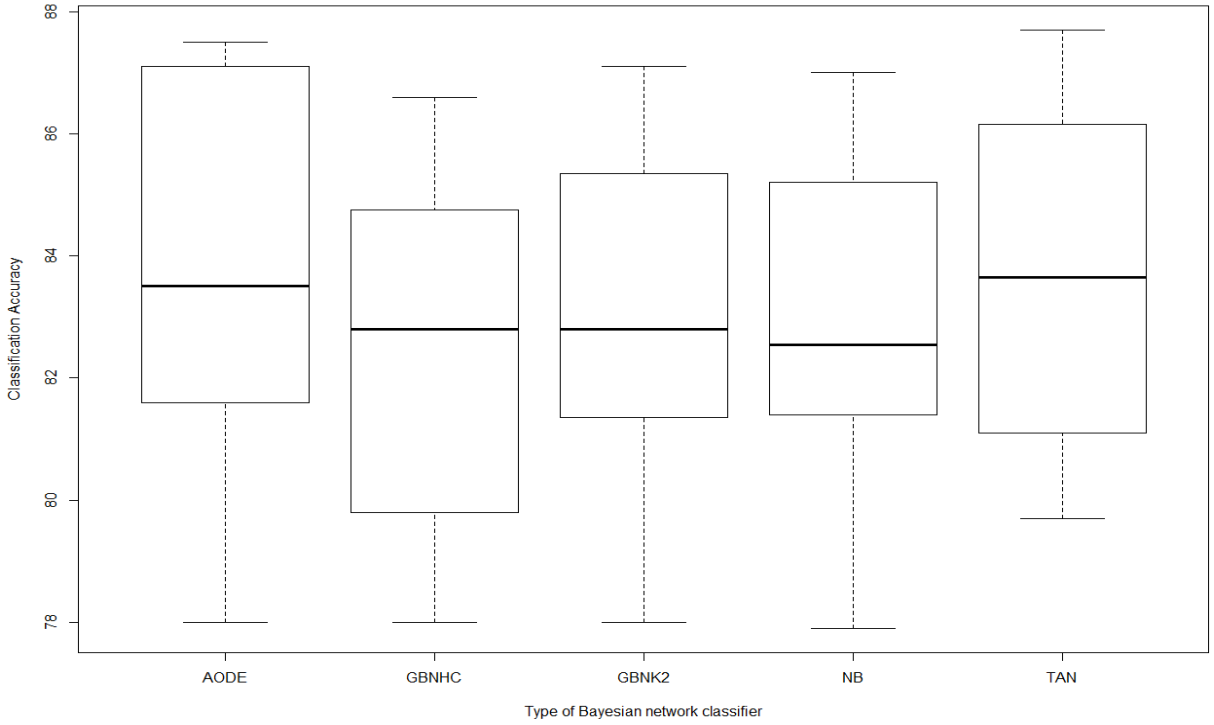
Appendix G



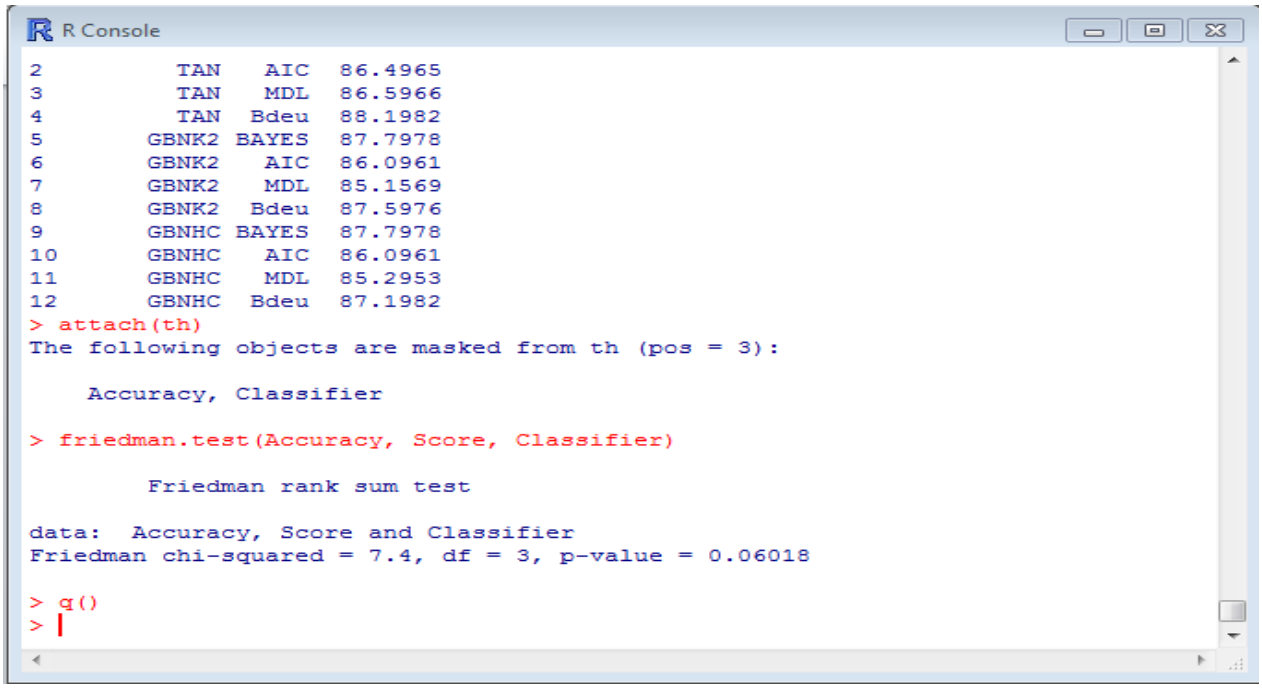
Appendix H

```
> require(PMCMR)
> posthoc.friedman.nemenyi.test(Accuracy, Classifier, Discretization)

        Pairwise comparisons using Nemenyi post-hoc test with q approximation for unreplicated blocked data

data:  Accuracy , Classifier and Discretization

      AODE GBNHC GBNK2 NB
GBNHC 0.15 -     -     -
GBNK2 0.97 0.46  -     -
NB    0.36 0.99  0.76  -
TAN   1.00 0.15  0.97  0.36

P value adjustment method: none
>
```

Appendix I



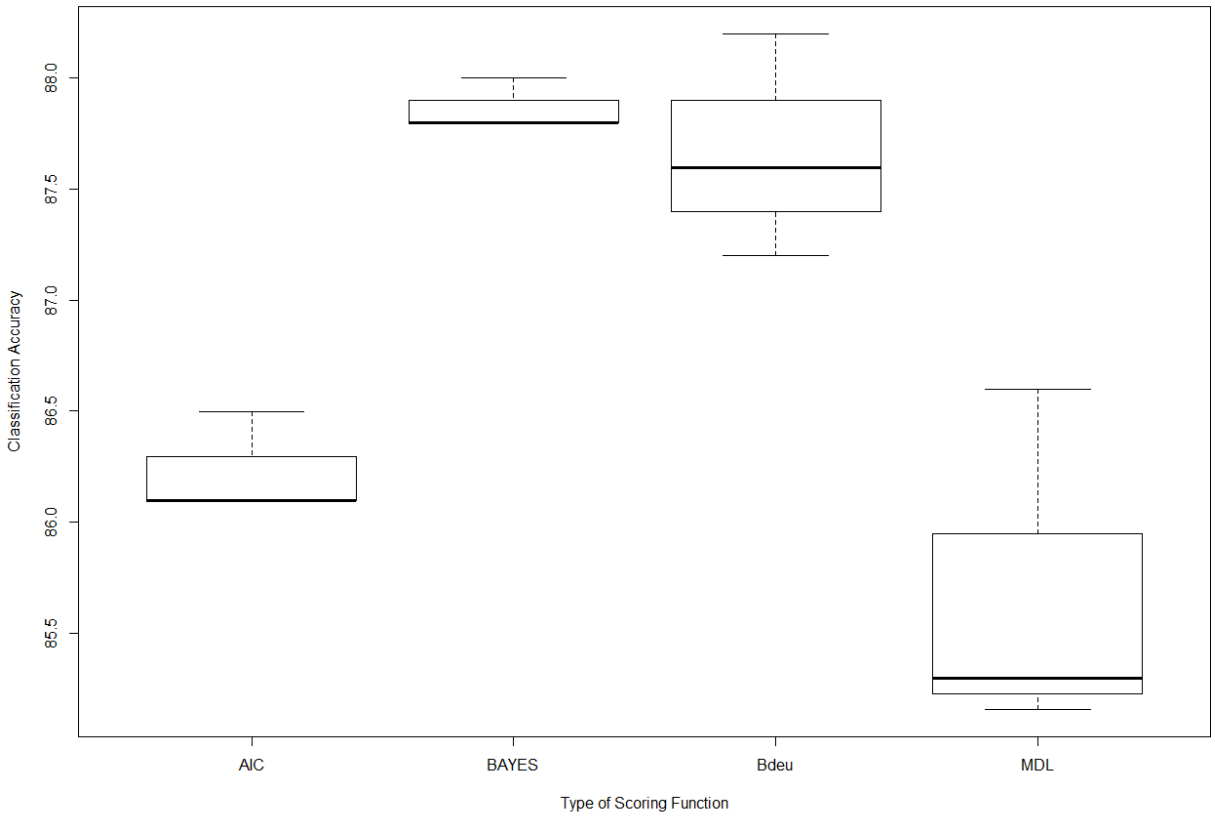
Appendix J



Appendix K

```
> require(PMCMR)
Loading required package: PMCMR
> posthoc.friedman.nemenyi.test(Accuracy, Score, Classifier)

        Pairwise comparisons using Nemenyi post-hoc test with q approximation f$

data:  Accuracy , Score and Classifier

      AIC  BAYES Bdeu
BAYES 0.23 -     -
Bdeu  0.39 0.99  -
MDL   0.99 0.12  0.23

P value adjustment method: none
>
```

Appendix M



Appendix N

NSL-KDD dataset – List of Features

Feature Set Description	Amount
Basic features of individual TCP	9
Content feature within a connection suggested by domain knowledge	13
Traffic features computed using a two-second time window	9
Host features	10
Total	41

NSL-KDD Dataset-Basic Features

Feature name	Description	Type
duration	length of the connection	continuous
protocol_type	type of the protocol e.g tcp, udp, etc	discrete
service	network service on the destination e.g http, telnet	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection id from/to the same host/port; o otherwise	discrete
wrong_fragment	number of "wrong"fragments	continuous
urgent	number of urgent packets	continuous

NSL-KDD Dataset – Content Features

Feature Name	Description	Type
hot	number of "hot" indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of "compromised" conditions	continuous
root_shell	1 if not shell is obtained; 0 otherwise	discrete
su_attempted	1 if "su_root" command attempted; 0 otherwise	discrete

num_root	number of "root" accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shells	continuous
num_access_files	number of shell prompts	continuous
num_outboundcmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise	discrete
is_guest_login	1 if the login is a "guest" login; 0 otherwise	discrete

NSL-KDD Dataset – Traffic features

Feature Name	**Description**	**Type**
count	number of conections to the same host as the current connection in the past two seconds	continuous
serror_rate	% of connections that have "syn" errors	continuous
rerror_rate	% of connections that have "rej" errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
srv_serror_rate	% of connections that have "syn" errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

NSL-KDD Dataset – Host Features

Feature Name	**Description**	**Type**
dst_host_count	number of connections to the destination host as the current connection in the past two	continuous
dst_host_srv_count	number of connections to the destination service as the	continuous

	current connection in the past two seconds	
dst_host_same_srv_rate	% of connections to the same service at destination host	continuous
dst_host_same_src_port_rate	% of connections to the same source ports at destination host	continuous
dst_host_srv_diff_host_rate	% of connections to the different host at destination host	continuous
dst_host_serror_rate	% of connections that have “syn” errors at destination host	continuous
dst_host_srv_serror_rate	% of connections that have “syn” errors at the destination host	continuous
dst_host_rerror_rate	% of connections that have “refj” errors destination host	continuous
dst_host_srv_rerror_rate	% of connections that have “ref” errors at destination host	continuous