A Model for Intrusion Detection in IoT using Machine Learning

By

Junior Ruddy Nkala

[BSc Honours]

A dissertation submitted in fulfilment 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 K. Sibanda

[February 2019]

Abstract

The Internet of Things is an open and comprehensive global network of intelligent objects that have the capacity to auto-organize, share information, data and resources. There are currently over a billion devices connected to the Internet, and this number increases by the day. While these devices make our life easier, safer and healthier, they are expanding the number of attack targets vulnerable to cyber-attacks from potential hackers and malicious software. Therefore, protecting these devices from adversaries and unauthorized access and modification is very important.

The purpose of this study is to develop a secure lightweight intrusion and anomaly detection model for IoT to help detect threats in the environment. We propose the use of data mining and machine learning algorithms as a classification technique for detecting abnormal or malicious traffic transmitted between devices due to potential attacks such as DoS, Man-In-Middle and Flooding attacks at the application level.

This study makes use of two robust machine learning algorithms, namely the C4.5 Decision Trees and K-means clustering to develop an anomaly detection model. MATLAB Math Simulator was used for implementation. The study conducts a series of experiments in detecting abnormal data and normal data in a dataset that contains gas concentration readings from a number of sensors deployed in an Italian city over a year. Thereafter we examined the classification performance in terms of accuracy of our proposed anomaly detection model.

Results drawn from the experiments conducted indicate that the size of the training sample improves classification ability of the proposed model. Our findings noted that the choice of discretization algorithm does matter in the quest for optimal classification performance. The proposed model proved accurate in detecting anomalies in IoT, and classifying between normal and abnormal data. The proposed model has a classification accuracy of 96.51% which proved to be higher compared to other algorithms such as the Naïve Bayes. The model proved to be lightweight and efficient in-terms of being faster at training and testing as compared to Artificial Neural Networks.

The conclusions drawn from this research are a perspective from a novice machine learning researcher with valuable recommendations that ensure optimal classification of normal and abnormal IoT data.

Statement of Original Authorship

I, Junior Ruddy Nkala 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 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: ……………………………………..;

Acknowledgements

Rejoice always, pray continually, give thanks in all circumstances; for this is God's will for you in Christ Jesus (1 Thessalonians 5:16-18). I would like to thank my omnipotent Father for his unfailing love, grace and providence that has guided me throughout this journey.

I am greatly indebted to my research supervisor Professor K. Sibanda for his immeasurable insight, encouragement, guidance and support he offered in this study. A special thanks to the Department of Computer Science for accepting me for the masters programme. Also a heartfelt thank you to my co- supervisor Mr Ngwenya for his motivation and concern throughout this study. Thank you to the Masters class of 2018 for the friendship, assistance and knowledge sharing. It was a worthwhile experience.

A special thank you, to my brother Sibhekisipho Fayayo, who encouraged me to apply for this programme, and to Andile Gama, Bekithemba Ndlovu, Manford Hlabano, Lloyd Dube, Nqobile Sikhosana, Prince Mahlangu, Advise Mpofu, Bongani Phiri, Ndumiso Nelela, and Dr Wayne Malinga, for their support and motivation. I will forever cherish all those unforgettable moments we had.

I would like to acknowledge the on-going support from parents Mr & Mrs R. Nkala, family members and the lovely Angela Nomvula Sithole, their encouragement, love, prayers and support through my academic pursuit was, and still is greatly appreciated.

List of Figures

List of Tables

List of Acronyms

IoT	Internet of Things
DoS	Denial of Service
DDoS	Distributed Denial of Service
ANN	Artificial Neural Network
IDS	Intrusion Detection System
NIDS	Network Intrusion Detection System
WEKA	Waikato Environment for Knowledge Analysis
GUI	Graphic User Interface
DT	Decision Tree
ML	Machine Learning
SVM	Support Vector Machines

TABLE OF CONTENTS

Chapter 1: Introduction

This study reviews the security and privacy challenges faced in the Internet of Things (IoT) implementations and architectures, while reviewing related work previously done by academia and scholars in an attempt to formulate methods for protecting the IoT from malicious attacks and cyber-security threats. The study then proposes a novel approach for anomaly detection based on Machine Learning and Data Mining as an Intrusion Detection Mechanism in IoT Systems. The purpose of this chapter is to provide underlying reasons for this research, it serves as an introduction to the chapters to follow and the research.

1.1 Background

The term Internet of Things (IoT) which is a system of interconnected devices was first proposed in 1999 by Kevin Ashton (Farooq et al., 2015) a co-founder of the Auto-ID Center at The Massachusetts Institute of Technology (MIT). IoT is basically a wireless interconnected network of a variety of objects such as Smart Phones, Sensors, Radio Frequency Identification Tags (RFID) and other types of wireless devices. It has application in areas such as health care, home automation, smart cities, modern agriculture and security just to name a few. IoT forms a global network infrastructure and is a self-configuring wireless network of sensors where the main goal is interconnectivity between various gadgets and objects (Vasilomanolakis, Daubert and Luthra, n.d.).

The Internet of Things from a different school of thought can be viewed as a worldwide network of uniquely addressable and interconnected objects, based on standard communication protocols. Devices that previously had no communication functions are being connected to a network by IoT Systems. These Systems enable discovery of phenomena that were previously unseen, providing new heights and insights. With the growth and advancements in RFID communications and Wireless sensor Networks over the last 10 – 15 years (Gartner, 2012), the feasibility of IoT and its various levels of architecture has also seen a parallel growth

With more and more devices and objects getting smarter there has been a rise in machine to machine communication, smart devices communicating with each other without the need for human interaction e.g. smart electricity meters that communicate their readings and other events directly to the server components over available mobile networks.

According to Gartner, around 25 billion uniquely identifiable objects are expected to be part of this global ecosystem by the year 2020, an impressively huge number, but the prevalence of such a big network will pose some new security and privacy threats and put all the devices at a high risk of hackers as they clutch at security gaps to make devices work to their own benefits (Farooq et al., 2015).

A previous study done by (Tanaka, Fujishima, Ohashi, & Tanaka, 2016) for the Hitachi Review documented IOT as a promising tool for efficiency and low costs or increasing sales, but however the IOT Acceleration Consortium (a collaborative program with members from industry, academia and the government) has underscored the need to handle three security issues (1) the increasing number of network–connected IoT devices, (2) long life cycles, and (3) the difficulty of perfect manual surveillance (Hitachi Review Vol 65, 2016).

The major security goals of IoT are to ensure proper authentication mechanisms and provide Data Availability, Confidentiality and Integrity (Farooq et al., 2015). IoT systems have however proven to be susceptible to various security attacks, such as denial-of-service (DoS) attacks and distributed denial-of-service (DDoS) attacks. Such attacks can cause considerable damage to the IoT services and smart environment applications in an IoT network. Consequently, securing IoT systems has become a major concern. For example, on Friday, October 21, 2016, a series of DDoS attacks were launched across the US that exploited the security vulnerabilities in IoT systems. These attacks affected IoT devices, websites and online services such as Twitter, Netflix, and PayPal. One solution that can be used to combat vulnerabilities in the IoT infrastructure and help prevent cyber-attacks such as DoS is the use of Intrusion Detection Systems. An Intrusion Detection System (IDS) is a security mechanism that works in real-time to analyse data being sent between two entities in a network and classifies the data in normal or abnormal data in an effort to secure the system from malicious entities. An IDS deployed for an IoT system should be able to analyze packets of data and generate responses in real time, analyze data packets in different layers of the IoT network with different protocol stacks, and adapt to different technologies in the IoT environment. An Intrusion Detection System that is designed for IoT-based system should operate under stringent conditions of low processing capability, fast response, and high-volume data processing. Therefore, conventional IDSs may not be fully suitable for IoT environments. IoT security is a continuous and serious issue; thus, an up-to-date understanding of the security vulnerabilities of IoT systems and the development of corresponding mitigation approaches are required.

1.2 Problem Statement

With the rapid increase of IoT devices being manufactured and deployed on to the internet, hackers have been targeting these relatively unsecure devices as source points to originate attacks such as the DoS and DDoS attacks. Most of these smart devices are deployed without security as a major goal and due to different security standards and interoperability differences they have proven vulnerable to hackers, as they can be used to launch ransomware attacks or act as a source of origin for Denial of Service attacks to attack larger sites (*IJACSA Vol 7, 2016).*

Security experts and Engineers constantly use different techniques and mechanisms to detect and mitigate threats such as DoS and DDoS which are broken down into four methods i.e. Statistical, Knowledge based, Soft Computing and Data Mining and Machine Learning methods (Munivara, Reddy and Rao, 2016).

This study argues that Machine learning techniques and methods are the best way to protect the IoT infrastructure from malicious threats by use of Intrusion Detection Methods that use significant patterns in traffic history and using that historical data to build efficient filtering rules that will be able to detect DoS and other attacks while securing the IoT network infrastructure. The first question that this study explores is;

How can we use Machine Learning techniques and algorithms to develop filtering rules that can help with Intrusion and Anomaly Detection in IoT networks?

Cyber-attacks vary in magnitude as some can take down a small intranet or a smart home, while others can render a nation-wide ISP provider unavailable, there have been attacks recorded of such magnitude, such as the Mirai DDoS attack of 2016 that peaked at 280 Gbps and rendered a number of internet sites unavailable, namely Netflix, Twitter and Facebook (Security Intelligence, 2017). It important to study how our solution will handle attacks of different seizes and magnitude. This leads to the second and final problem that this thesis seeks to investigate;

How resilient and accurate is our solution in detecting anomalies of varying magnitudes.

1.3 Aim and Objectives

The Aim of the study is to develop an effective and secure lightweight Intrusion and Anomaly Detection Model for IoT to help detect threats in the environment. This aim has been broken down to various research questions and objectives listed below:

- To review and study IoT Security Architecture requirements, goals and challenges.
- To investigate which Data mining, Machine learning algorithms and hypothesis that can be used to develop a comprehensive Anomaly detection Model for IoT.
- To develop an effective machine learning based model which can be for detection of anomalies and intrusions in an IoT infrastructure.
- To evaluate the proposed model.

1.4 Research Questions

1. What are the security challenges and vulnerabilities being faced in IoT security implementations and what tentative security measures can be used help improve security?
2. Can Data mining and Machine learning techniques be used as an effective way to develop a comprehensive defence mechanism against attacks on an IoT Network?
3. Which Machine learning and Data mining algorithms can be best implemented to effectively develop an efficient self-learning program/system to detect anomalies in data being transmitted in an IoT ecosystem.
4. How secure and accurate is the proposed intrusion and anomaly detection model.

1.5 Scope

In this study we discuss the security architecture of IoT and the security challenges and threats that it faces. The study is limited to designing and developing a model or framework for Anomaly and Intrusion Detection. This study is not focused on improving machine learning algorithms that can be used for threat detection and mitigation methods but rather to use data mining and machine learning techniques to develop a comprehensive defense mechanism against cyber-attacks on an IoT Infrastructure by analysing anomalies in data communicated between IoT devices. This study focuses on implementing a lightweight host-based intrusion detection model, which filters data by analysing anomalies in data at Application Level of the IoT architecture.

1.6 Justification

With the growth in the number of connected devices in the IoT ecosystem there has been a growth in the risk associated with the use of these devices. There will be an estimated 16 billion interconnected devices by 2020 (Kozlov, Veijalainen and Ali, 2012) therefore, the level of risk is only going to increase, hence there is a need to study at how we can improve the security architecture of the IoT and find new ways in which we can mitigate attacks. A Case Study of such an attack is a Denial Of Service Attack that occurred in October 2016 when an IoT Botnet known as the Mirai Malware generated a Flooding Attack through ordinary IoT gadgets such as Web Cameras, Smart Refrigerators and DVRs rendering a number of high profile websites such as Twitter and Netflix unavailable for a number of hours (Security Intelligence, 2017).

1.7 Methodology

There are 2 types of Research Methodologies, namely Qualitative and Quantitative, the latter is categorized into Experimental, Simulation and Inferential. Inferential approach entails forming a database from which to deduce the characteristics or relationships of the sample population (Silhavy et al., n.d.). Experimental involves conducting empirical studies with the aim of obtaining results from a real-world test-bed. Simulation is used for research that involves complex phenomena and hence cannot be setup in a laboratory environment. (Silhavy et al., n.d.)

For the purpose of this study we use a Simulation Approach as we are implementing a complex IoT Interconnected ecosystem and we do not have the tools to perform the study physically as with the experimental approach. We design and implement an Intrusion Detection Model using MATLAB Simulator, the model's purpose is to detect anomalies between data recorded and transmitted by sensor nodes in an IoT environment. The model must predict if certain data patterns are normal or abnormal. By analysing data patterns which usually change if there is an intrusion or when the system is under attack, as is common with DoS attacks and other malicious attacks, the system predicts whether there is an anomaly or normal data is being sent and received within the network.

1.8 Expected Results and Contributions

The study should reveal vulnerabilities and network security loopholes in IoT Architecture and in IoT implementations. The study shows the importance and benefits of securing IoT infrastructure and provide a platform for other researchers to further in this field of security in the field of IoT. The study provides a comprehensive research and analysis into data mining and machine learning techniques as a method to combat DoS, DDoS and any other malicious attacks on IoT Infrastructure and also at the end of the study contribute an efficient and comprehensive defence mechanism against the above mentioned threats.

1.9 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, research objectives and scope.

Chapter 2: This is a Literature Review, the chapter introduces The Internet of Things, the benefits of this relatively new landscape and its technologies. The chapter reviews the Security Aspects of the Internet, Cloud Computing and Cyber Security in general and later on focuses on IoT, how it is affecting cyber security and what are the threats and security challenges faced in IoT implementations. A special section is dedicated to Anomaly Detection techniques, as this is what the proposed model for intrusion detection will be based on.

Chapter 3: provides an in-depth literature review on Machine Learning algorithms, how they are implemented and how they can be used in an anomaly detection model to secure IoT architecture.

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

Chapter 5: This Chapter describes the results and presents the evaluations of the research. The aims of the study are stated, described and the results are analysed.

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

1.10 Conclusion

This chapter introduces the research study by providing background information on IoT and introducing the problem statement, listing the research objectives and outlines the research questions that are answered by this study. The chapter states the research scope and concludes with the dissertation structure. The chapter gives us a background of how this relatively new field IoT is using sophisticated technology to better the livelihoods of those using it, and how more processes are being automated, in the fields of Agriculture, Engineering, Medicine, etc. However, with the increased number in these IoT devices, there comes a threat posed by compatibility issues and lack of security which has seen IoT devices being used as entry points to launch certain cyber-attacks. Denial of Service and Distributed Denial of Service attacks have proven to be a threat in the past, hence there is a need for a comprehensive defense mechanism to protect IoT, this study proposes use of machine learning algorithms and data mining to develop an anomaly detection model that can be used as part of an Intrusion Detection System. An extensive review of IoT and the IoT Architecture follows in Chapter 2.

Chapter 2: The Internet of Things

This Chapter reviews the Internet of Things by introducing the notion of IoT and the underlying concepts of this technology. Furthermore it discusses the IoT Architectures and the Security Requirements, Goals and Issues related to this relatively new technology. The Chapter also discusses the Security Challenges faced in IoT implementations, and concludes by discussing Intrusion Detection and the threats posed by Network Intrusions on an IoT environment.

2.1 The Notion of Internet of Things

The term Internet of Things (IoT) was first coined at the Auto ID Centre of MIT by executive director Kevin Ashton in 1999, who used the term to describe a system of inter-connected devices (RFID Journal, 2009). The technology of IoT has so far caught the attention of society, industry and academy as a way of enhancing our daily activities (Ibarra-Esquer et al, 2017) and is becoming common in everyday use. The growth of IoT has been driven by the needs of large corporations that will benefit from the ability to code and track commodities and hence making companies more efficient, speed up processes, reduce errors, prevent theft and incorporate flexible organizational systems through IoT (Madakam et al, 2015). According to Madakam et al (2015), the Internet Of Things is a technological revolution that represents the future of computing and communications which combines a number of fields from wireless sensors to nano technology with every object in the ecosystem tagged and used to identify, automate and control.

Although the concept or idea of Internet of Things is over a decade old, the core technology behind this idea originates from the Internet as the term "Internet of Things" suggests. The Internet founded from project APRANET in 1969 by the US Defence Department, its objective being to link specialized computers to many general purpose computer centres for the department and some public and private sectors. Today, the Internet is a global system of interconnected networks that use TCP/IP protocols to serve billions worldwide (Madakam et al, 2015). It is a network that consists of government, schools, and businesses, private and public networks that are linked by an array of electronic, optical and wireless technologies.

Another technology behind the idea of IoT is the embedded computer system which is a term that was first coined in 1974, it is a term used to describe a small working system within a larger system with its sole purpose being that of performing a single task (Internet of Things

Agenda, 2009). These systems are implemented using microcontrollers and single board computers and have recently gained popularity as they are affordable and easy to use by prototyping platforms such as Arduino and Raspberry Pi.

In the early 1990s Mark Weiser proposed the concept of ubiquitous computing which is an idea that relies on advances in embedded computing technologies and deploying a ubiquitous network of the scale of hundreds of computers per room which resemble the Internet of Things, but at the time Weise argued that the main challenge was the design of an operating system that will allow software to take full advantage of the networking capabilities.

With the advances in wireless communications and digital electronics, sensor nodes started developing in the mid-90s. Sensors are capable of measuring and collecting data of their surroundings, what they are capable of measuring is broad and depends on the use and type of sensor, this ranges from GPS data, Temperature, Wind Speed, Acceleration, Proximity, etc. Sensor Nodes form part of the Wireless Sensor Network a key technology in IoT, sensor nodes record and transmit data over a network when required, and these sensors nodes are what are referred to as "things" or objects in IoT, the term things refers to a number of objects, both living and non-living such as a person, animal or a home appliance, mobile phone, sensors, etc.

2.2 Definition of IoT

According to Madakam et al (2015), there is no unique definition available for the Internet of Things that is accepted by all the community of users. They argue that there are many different groups of people including academics, researchers, practitioners, innovators, developers and cooperate people that have defined the term in the past. Although the term's initial use is accredited to Kevin Ashton, they argue that the best definition of IoT would be "An open and comprehensive network of intelligent tools and objects that have the capacity to auto-organise, share information, data and resources, reacting and acting in-face of situations and changes in the environment"

According to Zainab et al (2015), to understand IoT one has to go back to Mark Weiser's 1991 vision of the future internet under the name "Ubiquitous Computing", through his vision he focused on how one can turn a smart liveable environment in the presence of mobile phone technology.

Kevin Ashton versioned IoT as a system where through the use of RFID technology and Sensor Technology we can build an interconnected system of devices communicating with each other accomplishing a different number of tasks he wrote in a 1999 article for RFID Journal

"If we had computers that knew everything there was to know about things—using data they gathered without any help from us -- we would be able to track and count everything, and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best. We need to empower computers with their own means of gathering information, so they can see, hear and smell the world for themselves, in all its random glory. RFID and sensor technology enable computers to observe, identify and understand the world—without the limitations of human-entered data." – Ashton(1999).

According to Atzori et.al (2010), classified IoT into three paradigms, internet oriented, thing oriented and sematic oriented. Between 2008 and 2009 through the Cisco Internet Business Solutions Group IoT was defined as a set of smart objects such as home devices, mobile phones and appliances, addressed by a unique framework, this proposed framework was cloud computing.

The Internet of Things can also be considered as a global network which allows connection between human to human, human to things and things to things by providing a unique identity to each and every object communicating via the internet. In IoT sensors and actuators embedded in physical devices linked through wired and wireless networks using one IP address are connected to the internet. In IoT objects can sense the environment and communicate data and can internetwork with each other without human intervention. IoT refers to the coding and networking of everyday objects and things to render them individually machine-readable and traceable.

A more complete definition that encompasses most fields of IoT would be one given by Ron Van Kranenburg the founder of Council IoT in 2008 where he defined the Internet of Things as:-

"A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual 'Things' have

identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network" (Kranenburg, 2008).

2.3 IoT Architecture

According to Madakam et al.(2015), the problem with IoT is that it is vast and a broad concept with no uniform architecture, a typical idea/concept of IoT must consist of sensors, network communications and computer technology hence they have been many architectures and models given by several researchers and practitioners such as the European FP7 Research Project, IoT Forum Architecture, ITU Architecture and Qian Xiacong Zhang Architecture.

Farooq et al. (2015) argue that with an expected 25 Billion devices to be connected by the year 2020 which is a huge number, an existing architecture such as TCP/IP protocol cannot handle a network as huge as the IoT Ecosystem hence there is a need for a new open architecture that could address quality of service as well as various security concerns. Farooq seconds an architecture which consists of six layers suggested by Cheng, Zhang and Sun (2012) and uses the best features of the architectures of Internet and Telecommunications management networks based on TCP/IP and TMN models respectively, the layers in this architecture are the coding layer, network layer, middleware layer, application layer and business layer. However in its simplest form a basic IoT Architecture consists of 4 key layers described below:

- **Sensor Layer** - this is the lowest layer, consists of smart objects, allows interconnection of physical and digital world, allowing real-time information to be collected. They include sensors that can measure various data such temperature, GPS co-ordinates, Water levels and speed. In some cases, they may also have a degree of memory, enabling them to record a certain number of measurements. A sensor can measure the physical property and convert it into signal that can be understood by an instrument. Sensor layer also consists of actuators that can intervene to change the physical conditions that generate data e.g. cut power supply to a sensor recording wind speed.
- **Gateway/Network Layer** – Data recorded by Sensors will need to be transported to a remote server for processing, this requires an effective, efficient, robust and high performance network infrastructure that will operate as a transportation medium. With a wide range of IoT services and applications available such as transactional services, context aware applications, etc., multiple network solutions and various technologies can be used depending on requirements. Networks can be in form of public or private

models and are built to support various communication requirements such bandwidth and security. There are a number of Gateways available for use depending on the IoT function, these include but are not limited to microprocessors and microcontrollers, the networks that work hand in hand with these are GSM, GPRS, WSN, Wi-Fi, etc.

- **Support/Management Service Layer -** The Management Layer is where the processing of information occurs, through analytics, security control, process modelling and management of devices. The layer features business and process rule engines, which help in decision logic and trigger interactive and automated processes depending the data received from the sensor layer through the network/gateway layer. Data management is also done at this layer where techniques such as data anonymisation, integration and synchronization are used to hide details of the information while providing only essential information that is usable for the relevant applications.
- **Application Layer** – This layer represents the IoT applications that cover the "smart environments" in domains such as Supply Chain, Health care, Supply Chain and Energy.

A representation of IoT Architecture is given in Fig 2.1.

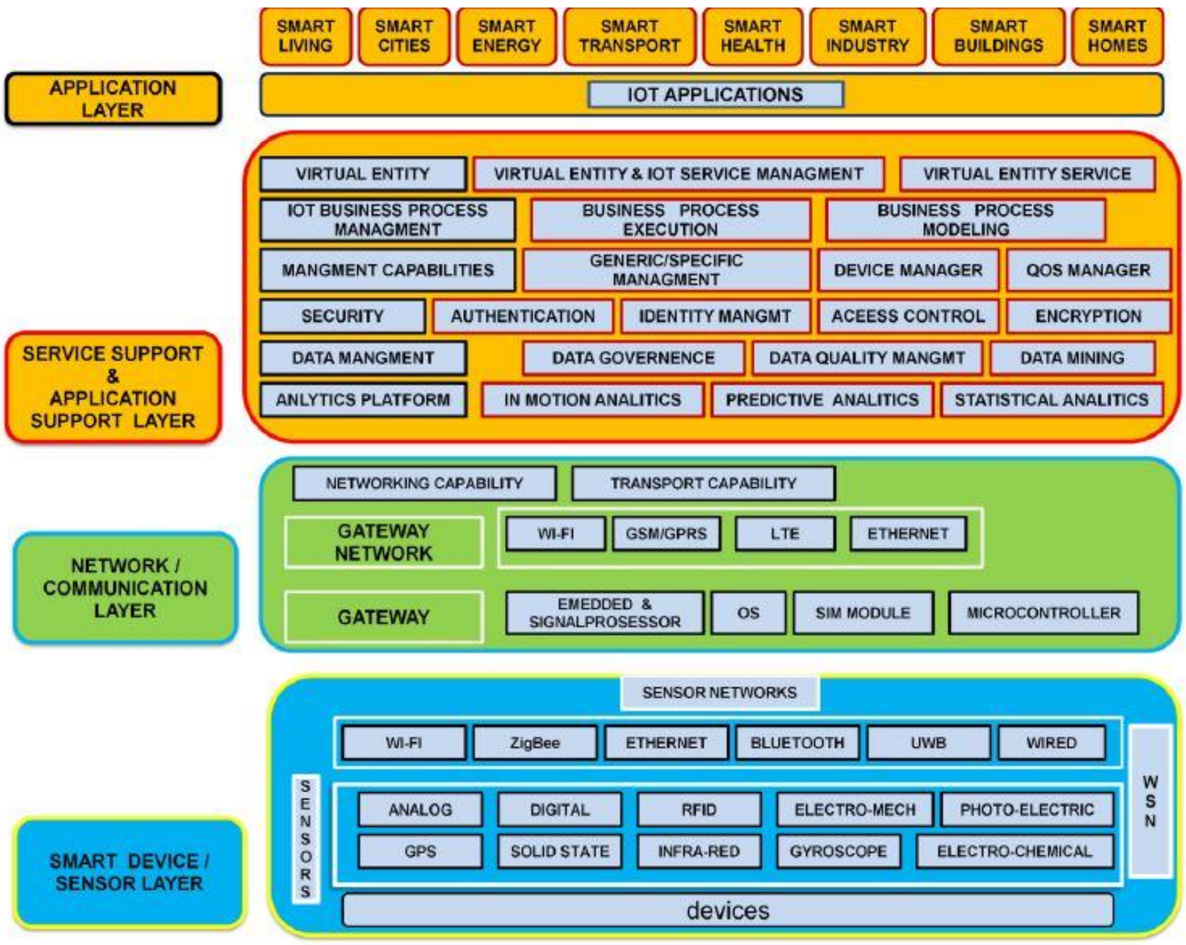


Fig 2.1 IoT Architecture (Source: Patel, 2016)

2.4 Technologies of IoT

Internet of Things consists of a number of elements and objects that exist in a ubiquitous computing system where each object can be uniquely identified and can interact with other objects to collect data based on which automated actions are taken, this is only possible with integration of new and effective technologies which are different but also relevant technologies. IoT was inspired by the RFID community and RFID technology remains the foundation and networking core of constructing the IoT. The Internet of Things includes technologies such as RFID, Sensor technology, nano technology, and intelligent embedded technology. IoT enables physical objects to be brought into the sphere of the cyber world, this is made possible by technologies such as Near Field Communication (NFC), RFID and 2D barcodes which allow objects to be identified and referred to the internet.

Below are some key technologies used in Internet of Things:-

- **Radio Frequency Identification (RFID)** – this is a system that transmits the identity of an object/person wirelessly using radio waves, it makes objects uniquely identifiable as each radio wave transmits a form of serial number (Madakam et al., 2015). It is a small transceiver microchip which can be integrated into a number of objects, it can both be active or passive. Active tags have a battery attached to them due to which they are always active and therefore continuously emit the data signals while Passive tags just get activated when they are triggered. Active tags are more costly than the Passive tags however they have a wide range of useful applications. The main components of RFID are tag, reader, antenna, access controller, software and server. It is more reliable, efficient, secured, inexpensive and accurate. RFID has an extensive range of wireless applications such as distribution, tracing, patient monitoring, military apps etc.
- **Internet Protocol (IP)** – this is the primary network protocol used on the internet, developed in the 1970s. IP is the principle communications protocol suite for TCP/IP, the two versions of Internet Protocol (IP) are in use IPv4 and IPv6, IPv6 is 21st century Internet Protocol, it supports around 2^{128} address (Bicknell, 2009).
- **Wireless Sensor Network** – is a bi-directional wirelessly connected network of sensors in a multi-hop fashion, it is built from a number of several nodes scattered in a sensor field each node connected to another sensor node which can collect object specific data such as temperature, acceleration, wind speed, etc. Sensor nodes may not have a global ID because of the large amount of overhead and large number of sensors. WSN based on IoT have made remarkable attention in areas such as military, healthcare, precision agriculture and manufacturing (Farooq, 2015).
- **Network Technologies** – Networking technologies play a very important role in IoT, they are the ones responsible for the connection and communication between object/nodes. Choosing a network transmission type depends on a number of factors such as efficiency, cost and distance. For wide-rage transmission networks. 3G and 4G are use as in mobile transmission, but for short range we might require a cheaper medium such as Bluetooth or Zigbee. There are a number of network technologies that IoT Systems and devices use, these include Wi-Fi, Bluetooth, 2G, 3G, 4G, Zigbee and NFC just to name a few.
- **Cloud Computing -** One cannot separate Cloud Computing and the Internet of Things, they both represent ubiquitous computing, and they both use the distributed computing

concept. According to Gartner (2012), 25 million device interconnected by the year 2020, the cloud seems to be a technology that can be the most effective solution to store, analyse and manage data for the Internet of Things (Farooq et al., 2015). Cloud Computing is considered a standard framework for IoT, integrating while IoT represents real world and small things, but it is limited storage in addition to traditional problems in the network such as scalability and privacy; in other side, cloud computing has virtually unlimited capabilities and processing power. Cloud computing interfaced with a large number of smart objects using millions of sensors can be a large benefit to IoT in large scale processing and analysing of data.

2.5 IoT Challenges

The Internet of Things (IoT) is a relatively new technology/field and as with most new technology, there exists limitations by which benefits can be achieved and a new approach may come with unique additional challenges. According to Jung, Cho & Kang (2014) IoT has a number of challenges, these include the fact that there is no standard architecture in IoT, vendors are keen on developing objects to achieve financial gain, but do not comply with other manufacture objects hence there is no set standard. They also mention that IoT Networks are heterogeneous, this makes managing IoT network security a tough task since these devices use different communication protocols over different types of network e.g. GSM, WAN and Bluetooth. The implementation of IoT is marred with critical issues such as questions over security and privacy, since the objects have limited hardware capabilities, it is almost an impossible task to deploy host based security mechanisms, and this is pushing developers towards anomaly detection and protection techniques.

2.5.1 Security in the Internet of Things

There are 3 main security goals in IoT, these are data confidentiality, availability and integrity. Unlike the Internet, IoT is to be applied at critical areas of a society, e.g. medical services, healthcare, military, and intelligent transportation hence the need for security as high availability and dependability is a must. The basic IoT security architecture is divided into 4 levels/layers and each level has its function. Figure 2.2 shows a typical IoT Security architecture and security requirements at each level.

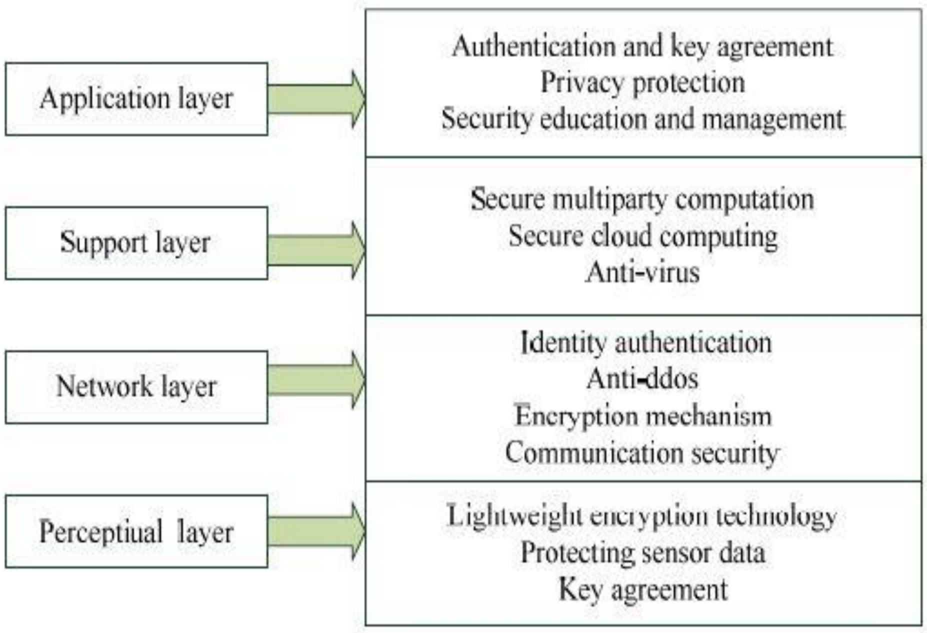


Fig 2.2 Security Requirements at each level (Source: Suo, 2012)

- **Sensor / Perceptual Layer** – this layer consists of sensors and computers with relatively lower power consumption and storage. It is difficult to set up security but attacks such as the denial of service can cause problems in this layer, hence the need to employ a defence mechanism. To protect against illegal access of these nodes or sensors in this layer and maintain the confidentiality of information being transmitted lightweight encryption methods such as Hash algorithms are used ensure that security is applied and performance and power consumption is not wasted by avoiding use of complex algorithms (Suo et al, 2012).
- **Network Layer** – this layer is vulnerable to attacks such as Man-In-The-Middle attack, and DOS attack, hence there is a need for a security mechanism at this level. There is need to establish confidentiality and integrity usually by use of complex algorithm and intrusion detection systems.
- **Middleware/Support Layer** – this layer uses technologies such as cloud computing and virtualization, both these are ripe to various attacks, hence there is a need for strong encryption algorithms and anti-virus systems and Intrusion Detection Systems. Mass data processing and intelligent decision is done at this layer hence there is a need to recognize information/data that might be malicious.
- **Application Layer** – in this layer, password management and access control are of importance, there is need for authentication at this layer to prevent issues such as wrong

disclosure of information, data privacy and access control are the main concerns in this layer.

2.6 Intrusion Detection in IoT

During a Network Intrusion such as Dos and Man in The Middle attacks, illegitimate traffic masks itself among the legitimate traffic to deplete some vital system resource and ultimately render the whole network or system unavailable. This makes it substantially harder to filter out the illegitimate traffic without substantially affecting the legitimate traffic. There are two strategies that can be adopted help detect Intrusions in a system, and help filter illegitimate traffic or requests from legitimate ones; these are namely

i. Anomaly detection
ii. Signature based detection

Anomaly Detection

Anomaly-based detection is basically concerned with identifying behaviours or events that might be anomalous with respect to normal behaviour (Modi et al., 2013). Anomaly-based detection techniques builds on the principle that traffic distribution of a service will change under an attack. Anomaly based detection first analyses traffic based on a traffic model the system defines as normal. The system will go through a training phase to define the normal behaviour before the system enters the detection phase. Since anomaly intrusion detection compares traffic against the normally defined behaviour, the system has the ability to detect unknown attacks. However, anomaly based detection depends on training and learning on a set of normal data. The problem lies in if that particular training data contains malicious software or content the system might be unable to detect these content when it enters the detection phase. There are a number of approaches inside of data collection, processing and filtering within anomaly detection techniques. Whereas, statistical analysis and machine learning are two of the more common approaches.

Signature Based Detection

Signature based detection techniques is able to detect attacks based on signatures of known attacks. It involves searching network traffic for a series of malicious bytes or packet sequences, it is easy to develop and understand if we know what network behaviour we want to identify. This approach is characterized by high detection rates and low false positives. A slight deviation or variation in the intrusive behaviour 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. The only problem with this approach is that it cannot detect unknown attacks, hence there is a need for constantly updating the signature base (Jyothsna & Prasad, 2011).

2.6.1 How do Network Intrusions pose a threat to IoT?

As stated earlier on in the chapter an intrusion is when legitimate access to a service or resources on a target node is disrupted by malicious and illegitimate requests or flow of data. Below we review how this form of cyber-attack can pose a threat to the IoT network:-

- The Internet is not designed to police intermediate traffic. Its end-to-end design make the intermediate network simple and optimized to ensure the fastest packet forwarding service while leave the complexity of packet processing to the hosts on the two ends of the communication. When proper detecting and preventive mechanism are missing on the receiver, the system becomes vulnerable to malicious packets streamed from the sender. In the IoT network, end devices are usually not equipped with high computational resources for implementing complex security algorithm and usually limited in power supply, which makes them not intelligent enough to detect and avoid network attack (Suo et al, 2012).
- The number of services available on one IoT network component is limited, which means only certain number of requests could be processed at a certain given time, one at a time. When malicious packets taking a large portion of the total requests, chances that legitimate requests being temporarily blocked becomes larger.
- IoT workflow is designed to be highly dependent on a number of connected devices over a network, a single point of failure would render the whole system unavailable and useless, for example, once DDoS attack brings down the serving device on a IoT network, the other IoT devices whose functions rely on the this blocked device will be also blocked from serving their client devices, which causes impairment of a local network (Zhang & Green, 2015).

2.7 Conclusion

This chapter gave an introduction to the Internet of Things and its technologies. The underlying concepts were stated and comprehensive definitions were provided. The essential characteristics, architectures and technologies were outlined. Security challenges and the vendor security mechanisms were discussed in this chapter. Finally, the chapter ended by discussing the issue of Intrusion detection in IoT. The information presented is essential in order to appreciate and understand the case-study area of this study.

Chapter 3: Machine Learning in IoT

The focal point of this study revolves around the concept of threat detection and mitigation in IoT Security Architectures by use of Machine Learning Algorithms. This Chapter introduces the sub-field of Computer Science, named Machine Learning. The Chapter reviews supervised and un-supervised methods of machine learning, a full literature review of various machine learning algorithms is presented including those that will be used in developing the proposed Anomaly Detection Model. In this chapter we review existing literature on data mining, and machine learning algorithms. Much attention is placed on the accuracy and performance of these particular algorithms. We carefully review previous work done by academia and industry in various case studies. The aim is to identify the approaches that will best suit this study.

3.1 Definition of Machine Learning

Machine Learning is a subfield of Computer Science and is a type of Artificial Intelligence that has evolved from a field of pattern recognition to computational theory (Mahdavinejad et al., 2018). It is used in the field of Computer Security to detect anomalies and changes in traffic and learn on making future predictions based on earlier observed data. Machine Learning provides Machines with the ability to learn without being explicitly programmed, to achieve this machine learning approaches go through a training phase before making decisions on new data.

Machine Learning is a field that has been around since the 1960s, Bell (2016) claims that Machine Learning was originally defined by Arthur Samuel an engineer at IBM as

"Field of study that gives computers the ability to learn without being explicitly programmed"

Tom Mitchell the chair of Machine Learning at Carnegie Mellon University, provided another definition of Machine Language in 1997 as:-

"A computer program is to learn from experience E with respect to some class of tasks T and performance P, if its performance at its tasks in T, as measured by P, improves with the experience E"

Mitchell's (1997) definition can be broken down as meaning, it can be said that a computer program that runs a set of tasks is learning if the performance on those particular tasks improves with experience. Therefore a system has the ability to accept inputs, process and analyse data systematically to find patterns and then undertake predictions is a system that is capable of learning, with more and more learning the predictions that the system can produce become more accurate.

There are three main categories of learning in Machine Learning, namely supervised, unsupervised and reinforcement training.

3.1.1 Learning in Machine Learning

According to Shalev-Shwartz and Ben-David (2014), although learning is important it is not the ultimate goal, the ultimate goal is to produce a system that can automatically and correctly detect meaningful patterns in data.

- **Supervised Learning** – in supervised learning labelled data is used the system is trained with labelled data, it (the system) is presented with labelled data in-order for it to build a knowledge base based on that particular data so it is able to recognize unlabelled data or data that does not exist in the built knowledge base. This would mean that the training phase should be able to determine if a certain packet A is either a normal packet or an abnormal packet. The system classifies new data based on known behaviour, inspect different attributes and then determine which category the packet falls into. The main objective of supervised learning is to learn how to predict the appropriate output vector from a given input vector (Mahdavinejad et al, 2018).
- **Unsupervised Learning** – unlike in supervised learning, this approach uses unlabelled data to learn normal data distribution. Unsupervised learning allows us to look at the data with little to none known idea of what the result should look like. This approach requires a machine to conduct systematic processing and analyse data, before breaking it down to groups with similar features based on attributes chosen previously. According to Chapelle (2006), it is assumed that data should follow a certain pattern or structure. Based on that particular structure we should be able to classify new data and detect anomalies in the data pattern.

3.2 Machine Learning Algorithms

A ML Algorithm is the driving force behind deciding which packets of data to accept or not to, in supervised and unsupervised learning (Kongshavn Rønning, 2017). An algorithm is a number of finite rules that have to be followed so as to solve a particular problem (Basu, 2013).

In Machine learning there are a number of algorithms which have similar functions, there can be split into a number of categories as mentioned in the sections that follow.

3.3 Classification Algorithms

Algorithms that implement classification are called classifiers. Classifiers can classify data into several pre-defined categories, but however it is normally done by classifying data into normal and abnormal. Classification will often run on a set of training data where the category is known, this is a form of supervised learning (Kongshavn Rønning, 2017). Below are some of the common and relevant classification techniques.

3.3.1 Naïve Bayes Algorithm

This is an algorithm that is bases itself on the Bayes theorem, for one to fully understand the algorithm they have to be familiar with the Bayes theorem and the probability theory.

Heckerman (1996) states that probability is the degree of belief in an event occurring 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. Frequentism interprets probability as a measure of the frequency of trials while Bayesianism views probability as a measure of the outcome of a trial.

Naïve Bayes model is a very simplified Bayesian probability model. The mode is based on assigning an event to a class that have the posterior probability. During training, the method stores a probalististic summary for each class, this summary contains the conditional probability of each attribute value, as well as the prior probability of the class. Each time the algorithm enters a new instance, it updates the probabilistic stored with that specific class. When the system now is given an unclassified object, the classifier uses a function to check which of the pre-defined classes the object is likely to belong to. Below is the function of the Naïve Bayes rule that is used:-

$$\boldsymbol{Probability = \frac{likelihood.prior}{evidence}} \tag{3.1}$$

Prior is the previous knowledge about that specific class. Given that in a training set of 70 fruits 30 are apple and 40 are bananas, it is more likely that the new fruit is a banana. This gives a prior probability of $\frac{30}{70}$ for an apple and prior probability of $\frac{40}{70}$ for a banana. The likelihood or probability supports that the next fruit might belong to a certain group. The algorithm can further be written in this way

$$\boldsymbol{P(B|A)} = \frac{\boldsymbol{P(A \,|\, B).P(B)}}{\boldsymbol{P(A)}} \tag{3.2}$$

Where;

P(B) is the probability or marginal probability of B

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

P(A | B) is the conditional probability of A given B.

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

Note: posterior probability is a probability value that has been revised by using additional information that is later obtained.

Bayes theorem is a description of an observer's belief of an event B is updated by occurrence of event A. Bayes theorem is used to compute inverse probability, which means that if P(B|A) is known, the probability of P(A|B) can be computed.

In Naïve Bayes an input vector of Z = (Z_1……, Z_m), Naïve Bayes classifies assume independence between feature and attributes of Z given the class variable t (Mahdavinejad et al, 2018). By use of Bayes' theorem we have

$$P(t = c|Z_1, \ldots\ldots\ldots, Z_m) = \frac{P(Z_1, \ldots\ldots.., Z_m \mid t = c)P(t = c)}{P(Z_1, \ldots\ldots\ldots.., Z_m)} \tag{3.2}$$

And by application of the independence assumption and some simplifications, we have

$$P(t = c|Z_1, \ldots\ldots\ldots\ldots. Z_m) \infty P(t = c) \prod_{j=1}^{M} P(Z_j|t = c) \tag{3.3}$$

Therefore, the form of the classification task is

$$y = arg\,max\,P(t = c) \prod_{j=1}^{M} P(Z_1|t = c) \tag{3.4}$$

Where y denotes the predicted class label for z. Different Naïve Bayes classifiers use different approaches and distributions to estimate $P(t = c)$ and$P(Z_j|t = c)$.

Naive Bayes classifications are often robust to attributes that are irrelevant as the classification mechanism takes into account evidence from many attributes to make the final verdict. The disadvantage of Naïve Bayes is that it requires a strong independent assumptions of data.

3.3.2 K-Nearest Neighbours

KNN is a case based supervised learning algorithm which stores all its training data for classification, it is used for classification and regression. The algorithm introduced by Dasarathy in 1991 , it is an algorithm that reads a set of labelled training set, and then it is used to classify an unlabelled testing set, and then it is used to classify an unlabelled testing set.(I. Hmeidi et al, 2008). In KNN, the aim is to classify a new dataset by looking at the “K” given data points in a training set that are the closest to it in the particular input space. To find K-nearest neighbours of the new dataset, the algorithm will use a distance metric, such as Euclidean distance, Mahalanobis distance or hamming distance. Given N training vectors, K-NN identifies the K nearest neighbours of any new vector. For example, if K is 3 and a new object is in the close proximity of two objects of class Y and one objects of class X, the object will be classified into belonging to class Y. An example of K-nearest neighbour classification can be seen in Fig 3.1, here two of the closed objects belong to the green class, while only one belongs to the red class hence the unknown object which is white will be classified into the green class.

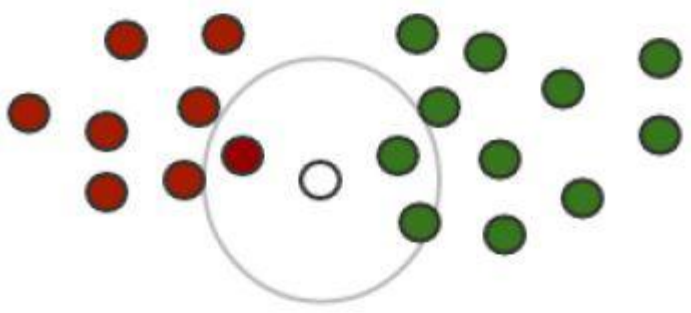


Fig 3.1 KNN Classification (Source: Kongshavn Rønning, 2017)

KNN is simple to implement, but its limitation is that it requires storing the entire training set, which makes it too costly. As a result of simplicity, certain issues besides storing the entire training set arise, that limit the performance of KNN for examples selecting the right distance measure, number of neighbours and larger part vote to combining class labels is not that effective.

KNN is implemented in different scenarios and utilized for a number of tasks such as Wireless Sensor Networks, Intrusion Detection for IoT systems and indoor positioning systems. KNN is robust simple as Naïve Bayes. To formulate the KNN Algorithm, let the input vector be x and its K nearest neighbour be N_k (x), the predicted label for x be y and the class variable be a discrete vendor variable of t

$$\boldsymbol{P(t = c|x, K) = \frac{1}{K} \sum_{i \in N_k(x)} 1(t_1 = c)} \tag{3.4}$$

$$\boldsymbol{y = arg_c max\, P(t = c|x, K)} \tag{3.5}$$

i.e., the input vector x will be labelled by the mode of its neighbours labels.

In K-NN, there are several ways to compute the amount of distance between data points, it all depends on the type of the features in the data. For real-valued features where ($\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{D}}$): Euclidean distance is commonly used (Mahdavinejad et al, 2018).

$$\boldsymbol{d(x_i, x_j)} = \sqrt{\sum_{\boldsymbol{m=1}}^{\boldsymbol{D}} (\boldsymbol{x_{im}} - \boldsymbol{x_{jm}})^2} = \sqrt{||\boldsymbol{x_i}||^2 + ||\boldsymbol{x_j}||^2 - 2\boldsymbol{x_i}^{\boldsymbol{T}}\boldsymbol{x_j}} \tag{3.6}$$

Simplification of the distance in-between points in 2 dimension.

$||\boldsymbol{x_i}|| = \sqrt{\sum_{\boldsymbol{m=1}}^{\boldsymbol{D}} \boldsymbol{x_{im}^2}}$ is the norm of x, or length of x.

$\boldsymbol{x_i^T x_j} = \sum_{\boldsymbol{m=1}}^{\boldsymbol{D}} \boldsymbol{x_{im} x_{jm}}$ is the dot (or inner) product of x_i and x_j

The dot product computes the similarity between two vectors (orthogonal vectors have dot product of 0, parallel vectors have a high dot product.

Hamming distances are binary valued, they count the number of features where the two examples disagree, so for hamming distance we use

$$\boldsymbol{d(x_i, x_j)} = \sum_{\boldsymbol{m=1}}^{\boldsymbol{D}} ||(\boldsymbol{x_{im}} \neq \boldsymbol{x_{jm}}) \tag{3.7}$$

3.3.3 Support Vector Machine Classifiers (SVM)

SVMs were introduced by Vladimir Vapnik, they use a supervised learning approach which can be used for both classification and regression tasks. It is a method of creating a set of functions from a set of labelled training data. SVMs are non-probalistic binary classifiers that separate both classes of the training set with maximum margin, then the predicted label of a new, unseen data point is determined based on which side of the hyperplane it falls.

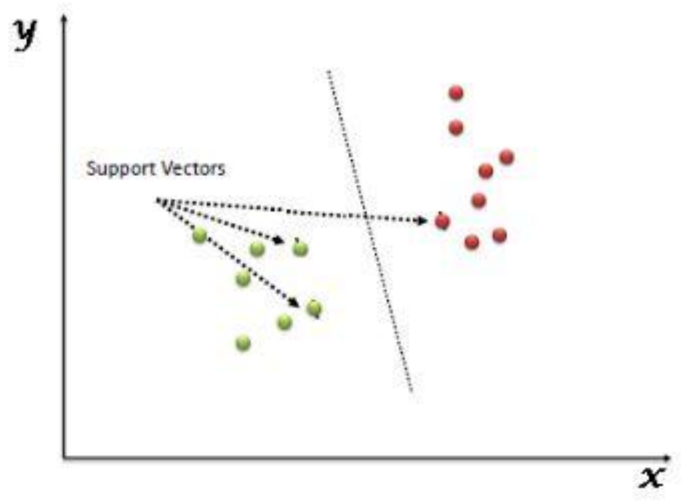


Fig 3.2 SVM Classification

In Fig 3.2 data is plotted in n-dimensional space (where n is the number of features we have) with each value of a particular coordinate. Then, we perform classification by finding the hyperplane that separates two particular classes.

Support Vectors are data points that lie close to the surface, they are data points that are difficult to classify, and they have a direct bearing on optimum location of the decision surface.

SVM looks at extremes of data sets and draws a decision boundary which is the hyperplane at near extreme points in the dataset (Danilo Bzdok et al, 2017). The question then lies in which hyperplane can be used to get the best results as shown in Fig 3.3.

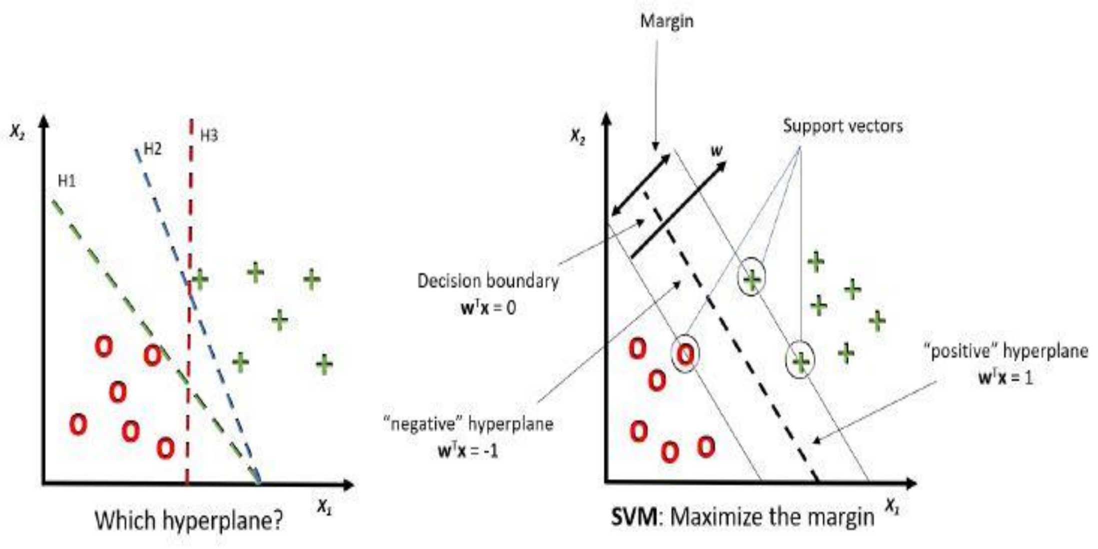


Fig 3.3 SVM Basic Concepts

SVMs work are used in datasets that can be linearly separated; give a dataset is not linearly separable, that particular dataset will be transformed into a higher dimensional space, so that a maximum-margin can be plotted. The problem with using this technique of transforming the dataset into a higher dimensional feature space is that it is expensive computationally. This particular problem can be avoided by applying what is called a "kernel trick" which would reduce the computational cost. The kernel trick or function makes use of a function that takes inputs as vectors in the original space and returns the dot product of the vectors in the feature. (Yazici, Basurra & Gaber, 2018). A disadvantage of Support Vector Machines is that they do not directly provide probability estimates, but they have proven to be versatile in specifying decision functions when using different kernels. They provide high precision, making them applicable to datasets with a large number of features, they are highly used in studying the air quality in urban areas of cities, image interpolation, as well as for medical classification which require low false positive and high precision rates.

3.4 Clustering Algorithms

In some cases it is not always possible to have sufficient knowledge about the data to efficiently classify new and unknown data, algorithms can use distance or similarity among data samples as a means to classify new data. Clustering is a method where data is divided into groups or clusters of similar objects. The purpose of clustering is to represent a set of unlabelled data into a set of natural hidden data structures. Most clustering algorithms require input of how many clusters you would want. Below are some of the clustering algorithms:-

3.4.1 K-Means Algorithm

This algorithm aims to divide N observations in K clusters or groups, where each observation belongs to a cluster with the nearest mean. This approach can be used with KNN classification, where K-means clustering is used to obtain clusters before KNN is used to classify new and unknown data from already existing clusters (Kongshavn Rønning, 2017).

K-means is used as an iterative solution to find the local minimal solution. This algorithm can be called the K-means algorithm or Lloyd’s Algorithm. However, Lloyd’s algorithm is based on the observation that the optimal placement of a mean is at the centroid of the associated cluster. The approach begins with choosing the different cluster centres or means. When the cluster centers have been chosen, the algorithm follows two steps; the first step determines which data belongs to which cluster via nearest distance calculation from the different points to the different means. The position of the cluster centers is then recomputed and moved based on finding the nearest center from all points in a cluster. The K-means algorithm will follow these two steps until convergence is reached. Even though this technique is able to find the local minimal solution, it’s not necessary the global minimal solutions, as the Lloyd’s algorithm does not specify the initial starting placement of the clustering centers. This is a serious weakness as the iterative technique is sensitive to the initial starting positions of the cluster centers. In other terms, how well the clustering is, heavily depends on where the initial cluster centers are set. There is currently no known efficient and widely accepted solutions to this problem. However, in order to obtain optimal clusters or solutions using the k-means algorithm, the algorithm is often ran several times with different starting positions for the cluster centers (Mahdavinejad et al , 2018). However, it is important to acknowledge that this is not an ideal solutions and several other techniques have been proposed. Among these solutions is an approach that tries to find a better starting condition so the algorithm can converge to a better local minimal. This can be done by trying to calculate different vector areas where the density is strongest, before setting these areas as the starting positions.

3.4.2 Density-based Spatial Clustering of applications with Noise (DBSCAN)

DBSCAN is a density-clustering algorithm which builds clusters based on points that are closely linked together, the purpose of the DBSCAN is to group a given unlabelled data set based on the density of its data points. Using this particular algorithm in a model, groups of dense data points (data points with many close neighbours) are considered as clusters and data points in regions with low-density are considered as outliers (Kriegel et al, 2011). The algorithm takes two parameters; *minpts* and distance *d*. If a point *Y* can reach *minpts* in a radius, based on distance *d*, point *Y* is considered a core-point. If point *Y* then have a path p_1, p_2,…………..,p_{n-1}, p_n to point *N*, where each point between point *Y* and point *N* is a core-point, all points p_x on the path needs to be density-reachable to p_{x+1}. Density-reachable means p_x need to have p_{x+1} within distance *d*. These core-points on the path p_1,p_2,…………..,p_{n-1}, p_n, including the points that is density-reachable to a core-point in this path, is considered as a cluster. Points that are density-reachable to a core-point, but is not considered as a core-point because it doesn't contain *minpts* within radius *d* are considered outliers of a cluster.

DBSCAN is a popular density-based clustering algorithm which allows clusters to expand in any shape and form. Unlike K-means, which assume all points in a dataset are legitimate, DBSCAN is resistant against noise, as any point which can't satisfy the *minpts* criteria, and is not connected to a core-point, is considered as an outlier in the dataset. Since DBSCAN builds clusters based on points that are linked together, the algorithm doesn't need to define the amount of start-clusters. However, since the clustering algorithm needs to define *minpts* and distance d this can cause several weaknesses. Either, if the dataset are not understood correctly to choose meaningful parameters, or that it is impossible to cluster data with a high differences in densities (Kongshavn Rønning, 2017).

In practice, DBSCAN is mostly used in large datasets as it is efficient, fast and robust against outliers. Although it has a significant advantage over other clustering methods the DBSCAN in the case of a data set with large differences in densities, the resulting clusters are destitute. DBSCAN is one of the most used clustering algorithms and is used in anomaly detection in temperature data and X-Ray crystallography (Mahdavinejad et al, 2018).

3.5 Decision Tree Algorithms (DT)

DT algorithms are known for their use of no parameters, comprehensibility, being able to handle mixed-type data and simplicity. A DT is induced from a set of labelled training data represented by a tuple of attribute values and a class label. Therefore due to the vast search space, the training process is typically a top-down, greedy and recursive process starting with the entire training data and an empty tree. An attribute that best partitions the training data is chosen as the splitting attribute for the root, and the training data are then partitioned into disjoint subsets satisfying the values of the splitting attribute. For each subset, the algorithm proceeds recursively until all instances in a subset belong to the same class. Decision Trees learning works considerable well on large datasets, the decision-tree algorithm has been proven to considerable outperform the Naive Bayes algorithm on larger datasets, while Naïve Bayes performs better than it on smaller datasets (Kohavi 1996; Domingos & Pazzani 1997).

Fig 3.4 shows an example of a Decision Tree

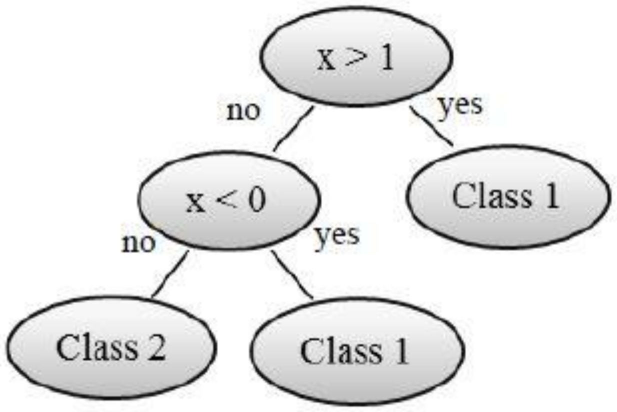


Fig 3.4 A simple Decision Tree

A Decision tree is constructed in two stages:

1. A growth stage
2. A prune stage

The process of construction the preliminary tree is called the 'growth stage', after the preliminary tree has been built, a sub-tree is created with the least estimated error rate, this process is known as the 'prune stage'. The pruning process consists of removing small, deep nodes from the preliminary tree resulting from 'noise' contained in the training sample that reducing the risk of overfitting and ensuring more precise classification of unknown data (Mahdavinejad et al , 2018).

There are a number of variants of the decision tree classifier/ algorithm, these include:-

- ID3 Algorithm
- M5 Algorithm

- C 4.5 and 5.0 Algorithm
- CART Algorithm

3.5.1 Iterative Dichotomiser 3 (ID3) Decision Tree Algorithm

ID3 is a supervised learning algorithm created by Ross Quinlan at the University of Sydney, which is used to form a decision tree by calculating the entropy values. For one to understand the ID3 algorithm, they have an understanding of the Theories of Shannon which is at the base of ID3 and C4.5 algorithms. Shannon Entropy first defines the amount of input information provided in an event, when a low-probability event occurs, the event carries more information that when the event has a higher-probability value (Badr et al, 2013).

Shannon Entropy

If we are given a probability distribution P = $(p_1, p_2 \ldots\ldots p_n)$ and a sample S then the Information carried by this distribution, also called the entropy of P is giving by:

$$\boldsymbol{Entropy(P) = -\sum_{i=1}^{n} p_i \times \log p_i} \tag{3.8}$$

Information gain G (p, T)

Entropy is a degree of randomness of data. It is used to calculate homogeneity of a data attribute. If Entropy is zero then the sample is totally homogeneous and if it is one then the sample is completely uncertain.

Information Gain or Gain is a decrease in entropy.

There are functions that allow us to measure the degree of mixing classes for all sample and any position of the tree in construction. It remains to define a function to select the test that must label the current node.

It defines the gain for a test T and a position p

$$Gain(p,T) = Entropy(p) - \sum_{j=1}^{n}(p_j \times Entropy(p_j)) \quad (3.9)$$

Where values (p_j) is the set of all possible values for attribute T. We can use this measure to rank attributes and build the decision tree where at each node is located the attributes with the highest information gain among attributes not yet considered in the path from the root.

The term dichotomisation means dividing into two completely opposite things, hence the algorithm iteratively divides attributes into two different groups which are the most dominant attribute to construct a tree, then entropy and gain is calculated for each attribute, after that the most dominant attributes are put up on the tree as a decision node. After that the entropy and gain of the remaining attributes are calculated among the remaining attributes and the next dominant attribute is found (Badr et al, 2013). This process is done until reaching a decision for that particular branch, the process is iterative hence the term "Iterative Dichotomisation".

```
Algorithm 1: ID3 Algorithm
Inputs: R: a set of non- target attributes,
        C: the target attribute,
        S: training data.

Output: returns a decision tree
Start
Initialize to empty tree;
    If S is empty then
        Return a single node failure value
    End If
    If S is made only for the values of the same target then
        Return a single node of this value
    End if
    If R is empty then
        Return a single node with value as the most common value of the target attribute
        values found in S
    End if

D ← the attribute that has the largest Gain (D, S) among all the attributes of R
{$d_j$ j = 1, 2, ..., m} ← Attribute values of D
{$S_j$ with j = 1, 2, ..., m} ←The subsets of S respectively constituted of $d_j$ records attribute value D

Return a tree whose root is D and the arcs are labelled by $d_1$, $d_2$, ..., $d_m$ and going to sub-trees ID3 (R-{D}, C,
S1), ID3 (R-{D} C, S2), .., ID3 (R-{D}, C, $S_m$)

End
```

Table 3.1 describes weather conditions that will determine the suitability of playing tennis outside for the previous 14 days.

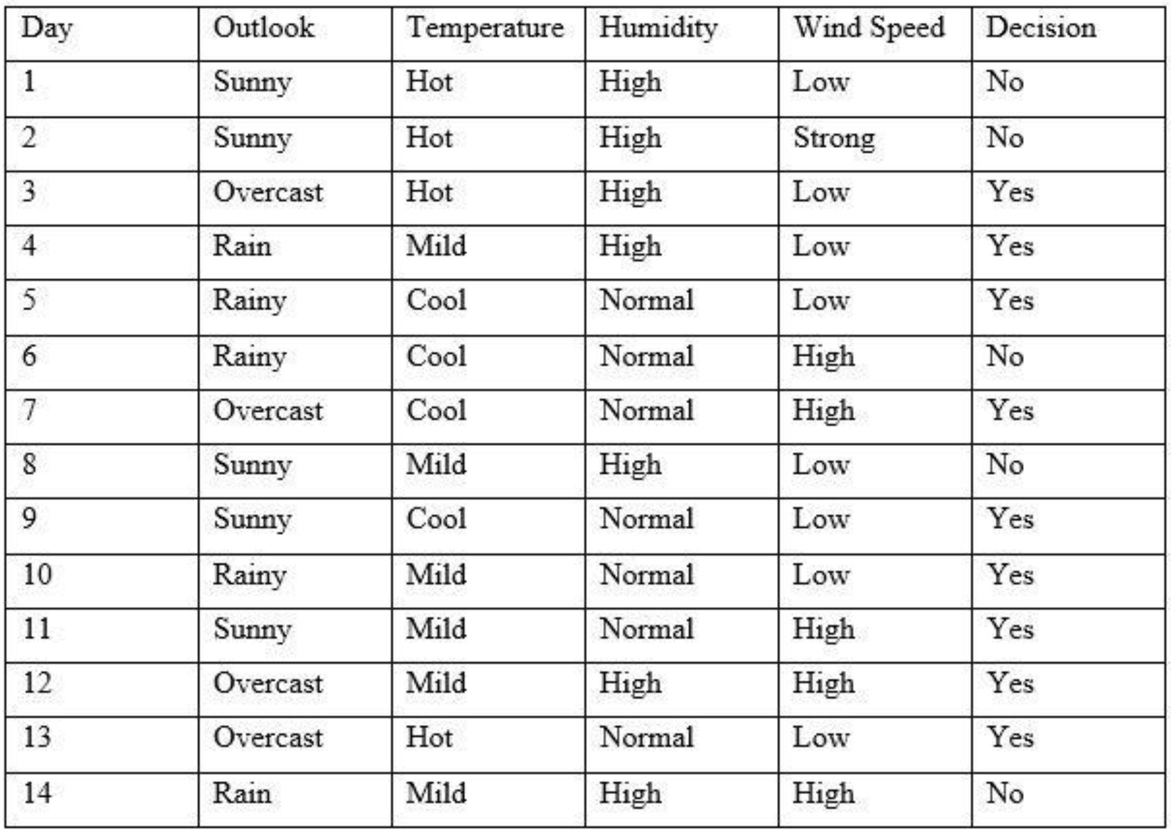
In the example, the classification target is “Should we Play Tennis?” which can be a “Yes” or a “No”. Weather Attributes are outlook, temperature, humidity and wind speed. The weather attributes have the following values:-

- Outlook = {Sunny, Rainy, Overcast}
- Temperature= {Hot, Mild, Cool}
- Humidity ={High, Normal}

- Wind={High, Low}

Examples of the Set S are

Table 3.1 Dataset S



Day	Outlook	Temperature	Humidity	Wind Speed	Decision
1	Sunny	Hot	High	Low	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Low	Yes
4	Rain	Mild	High	Low	Yes
5	Rainy	Cool	Normal	Low	Yes
6	Rainy	Cool	Normal	High	No
7	Overcast	Cool	Normal	High	Yes
8	Sunny	Mild	High	Low	No
9	Sunny	Cool	Normal	Low	Yes
10	Rainy	Mild	Normal	Low	Yes
11	Sunny	Mild	Normal	High	Yes
12	Overcast	Mild	High	High	Yes
13	Overcast	Hot	Normal	Low	Yes
14	Rain	Mild	High	High	No

We need to find the attribute that will be the root node in our decision tree, to find that we first have to calculate our entropy. The decision column consists of 14 instances and includes values of yes or no. There are 9 decisions labelled “yes”, and 5 decisions labelled “no”

$$\boldsymbol{Entropy(S) = -\sum_{i=1}^{n} p_i \times \log p_i}$$

Entropy(Decision) = - p(Yes).$\log_2$p(yes) – p(no).$\log_2$p(no)

Entropy(Decision) = -(9/14).$\log_2$(9/14) – (5/14).$\log_2$(5/14) = 0.94

To calculate the dominating attribute

We take one of the Attributes and calculate its gain.

$$\boldsymbol{Gain(p,T) = Entropy(p) - \sum_{j=1}^{n}(p_j \times Entropy(p_j))}$$

Wind Factor on decision.

Gain(Decision, Wind) = Entropy(Decision) – ∑ [p(Decision|Wind) . Entropy(Decision|Wind)]

Wind attribute has two labels: low and high. We would reflect it to the formula.

Gain(Decision, Wind) = Entropy(Decision) – [p(Decision|Wind=Low).

Entropy(Decision|Wind=Low)]-[p(Decision|Wind=High).Entropy(Decision|Wind=High)

Now, we need to calculate (Decision|Wind=Weak) and (Decision|Wind=High) respectively.

There are 8 instances for low wind, there 2 "No" decisions and 6 "Yes" decisions.

Entropy(Decision|Wind=Low) = -p(2/8)*log$_2$(2/8) – (6/8)*log$_2$(6/8) = 0.811

There are 6 instances for High wind. Decision is 3 on each side.

Entropy(Decision|Wind=High) = -p(3/6)*log$_2$(3/6) – (3/6)*log$_2$(3/6) = 1

Gain(Decision, Wind) = 0.94-[(8/14)*0,811]-[(6/14)*1] = 0.048

The same calculations are performed for the remainder of the attributes.

Gain(S, Outlook) = Entropy (S) - 5/14*Entropy (S|Sun)

- 4/14*Entropy (S|Rain)

- 5/14* Entropy (S|Overcast)

= 0.94 – 5/14*0.9710-4/14*0 – 5/14*0.9710

Gain(S, Outlook) = 0 .246

Calculation of entropies:

Entropy (S|Sun) = -2/5*log2 (2/5)-3/5* log2 (3/5) = 0.9710

Entropy (S|Rain) = -4/4*log2 (4/4)-0* log2 (0) =0

Entropy (S|Overcast) = -3/5* log2 (3/5) -/5* log2 (2/5) =0.9710

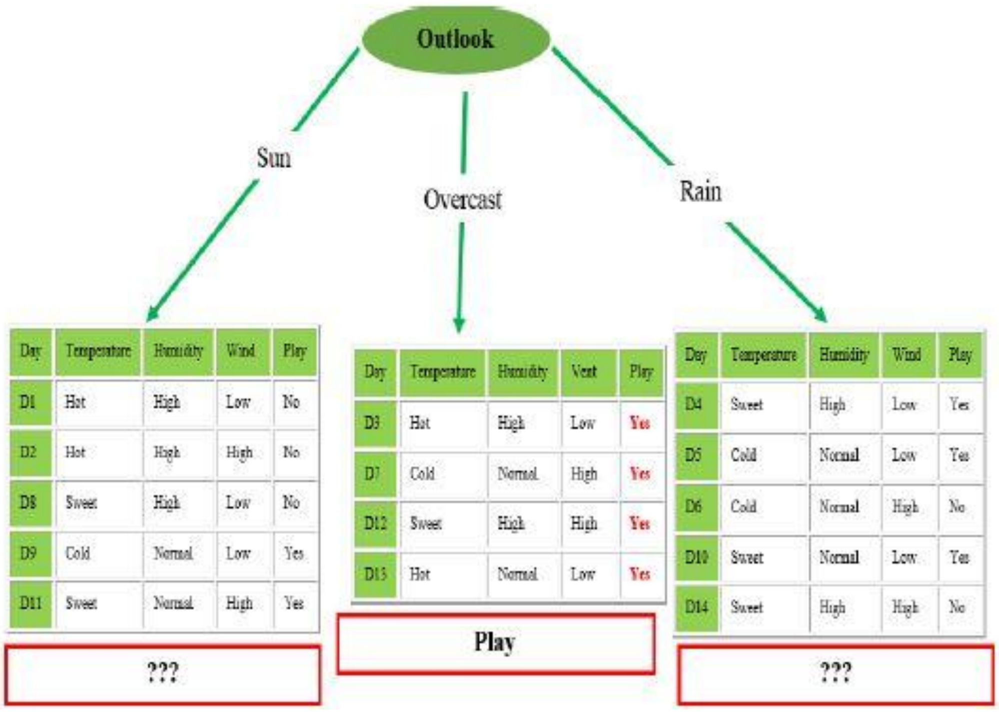
As well we find for the other variables:

Gain(S, Wind) = 0.048

Gain(S, Temperature) = 0.0289

Gain(S, Humidity) = 0.1515

Outlook attribute has the highest gain, so it is used as a decision attribute as the root of the tree. As show below



Day	Temperature	Humidity	Wind	Play
D1	Hot	High	Low	No
D2	Hot	High	High	No
D8	Sweet	High	Low	No
D9	Cold	Normal	Low	Yes
D11	Sweet	Normal	High	Yes

Day	Temperature	Humidity	Vent	Play
D3	Hot	High	Low	Yes
D7	Cold	Normal	High	Yes
D12	Sweet	High	High	Yes
D13	Hot	Normal	Low	Yes

Day	Temperature	Humidity	Wind	Play
D4	Sweet	High	Low	Yes
D5	Cold	Normal	Low	Yes
D6	Cold	Normal	High	No
D10	Sweet	Normal	Low	Yes
D14	Sweet	High	High	No

Fig 3.5 ID3 Tree Construction

By continuous iterations, the final tree is shown below in Fig 3.6

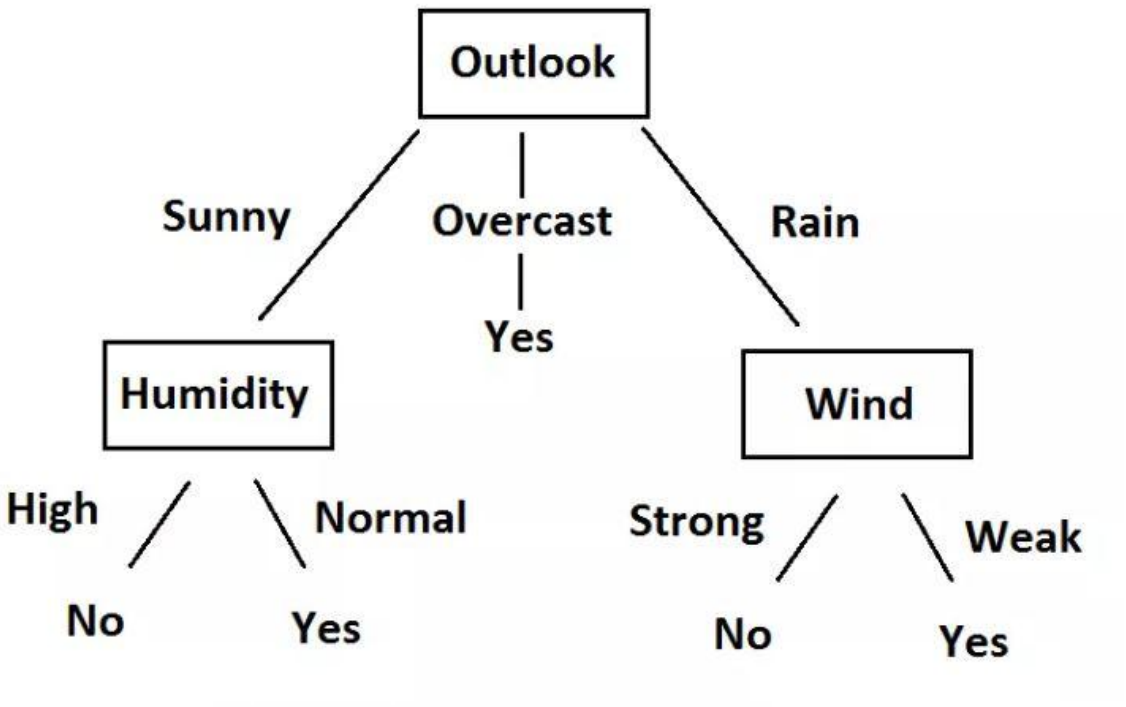


Fig 3.6 Complete ID3 Tree

3.5.2 C4.5 and 5.0 Algorithms

The C 4.5 algorithm introduced by Ross Quinlan is an extension to the ID3 algorithm, it acts as a solution to a shortfall by the ID3 Algorithm. As effective a classification algorithm as the ID3 algorithm is, it has a limitation in that it is overly sensitive to features with large numbers of value. Attributes in the ID3 must always be nominal values, the dataset must not include missing data and the algorithm tends to fall into overfitting (Mahdavinejad et al, 2018).

To overcome this problem, the C4.5 uses "Information Gain", which allows the algorithm to create a more generalized model including continuous data and can handle missing through measuring of gain ratio.

$$\boldsymbol{GainRatio(p,T)} = \frac{\boldsymbol{Gain(p,T)}}{\boldsymbol{SplitInfo(p,T)}} \tag{3.10}$$

Where SplitInfo is:

$$\boldsymbol{SplitInfo(p, test)} = -\sum_{j=1}^{n} P'\left(\frac{j}{p}\right) \times \log\left(P'\left(\frac{j}{p}\right)\right) \tag{3.11}$$

$P'(\frac{j}{p})$ is the proportion of elements present at the position p, taking the value of j-th test. Unlike Entropy, the definition is not dependent of the distribution of examples inside the different classes. Like in ID3 data is sorted at every node of the tree in order to determine which is the best attribute for spliting. Gain ratio impurity method is used to evaluate the splitting attribute (Quinlan, 1993).

On Attributes of Unknown Value

During the construction phase of the tree, it is possible to handle data for attributes that have an unknown value by evaluating the gain or the gain ratio. By use of a decision tree it is possible to classify the records that have unknown values by estimating the probabilities of the outcome.

The new criterion gain will be of the form:

$$\boldsymbol{Gain(p)} = F\left(\boldsymbol{Info(T)} - \boldsymbol{Info(p, T)}\right) \tag{3.12}$$

Where

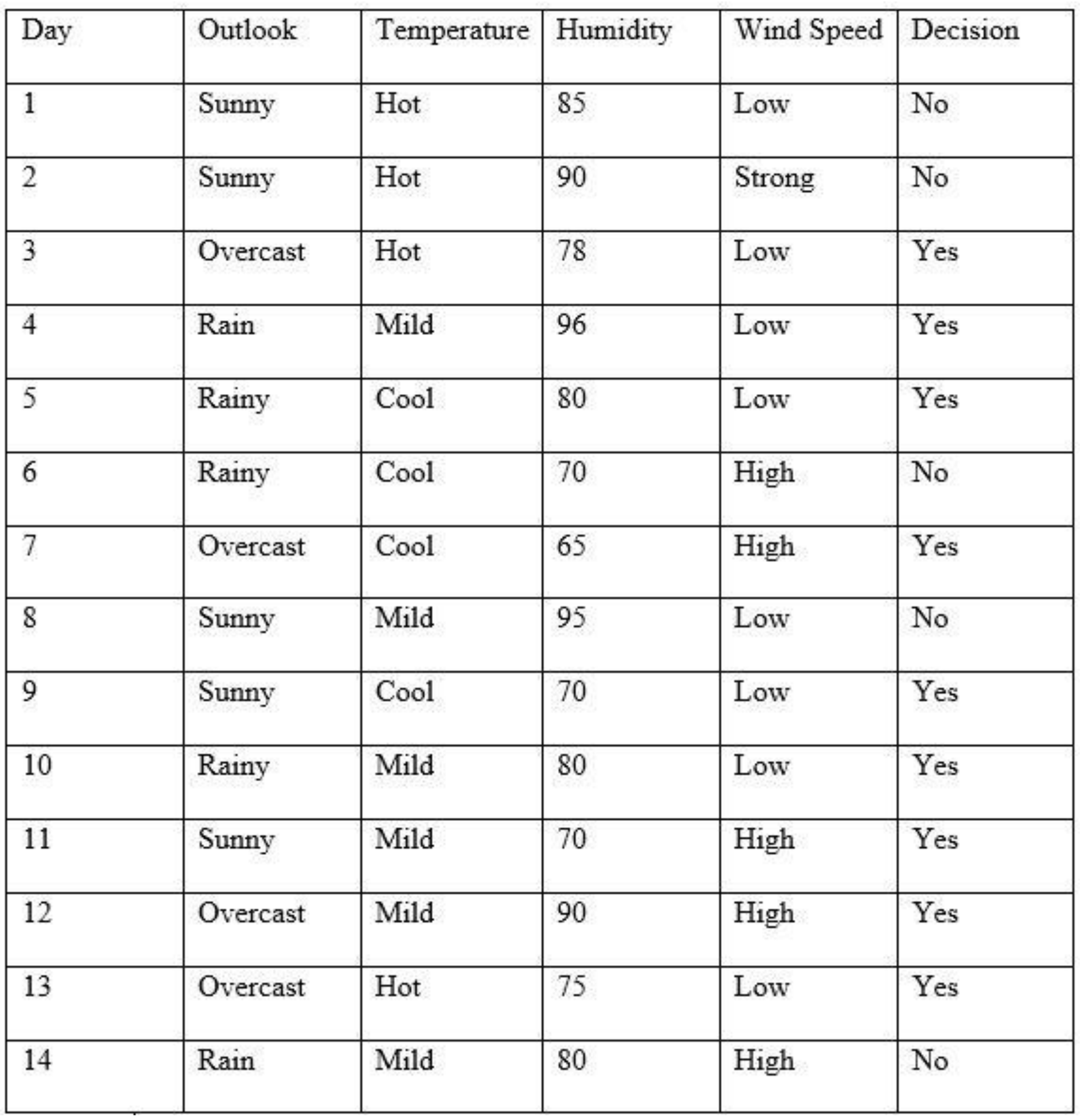
$$\boldsymbol{Info(p, T)} = \sum_{j=1}^{n} \left(p_j \times \boldsymbol{Entropy}(p_j)\right) \tag{3.13}$$

Info(T) = Entropy(T)

F is the number of samples in the dataset with a known value for a given or total number of samples in a set of attribute data.

Below is a training set similar to the one used for the ID3 example above, but this consists of consists of continuous values for Humidity instead of “High or Low”.

Table 3.2 Training Set with Continuous Values



Day	Outlook	Temperature	Humidity	Wind Speed	Decision
1	Sunny	Hot	85	Low	No
2	Sunny	Hot	90	Strong	No
3	Overcast	Hot	78	Low	Yes
4	Rain	Mild	96	Low	Yes
5	Rainy	Cool	80	Low	Yes
6	Rainy	Cool	70	High	No
7	Overcast	Cool	65	High	Yes
8	Sunny	Mild	95	Low	No
9	Sunny	Cool	70	Low	Yes
10	Rainy	Mild	80	Low	Yes
11	Sunny	Mild	70	High	Yes
12	Overcast	Mild	90	High	Yes
13	Overcast	Hot	75	Low	Yes
14	Rain	Mild	80	High	No

Gain is calculated as in the ID3 example except for the attributes with continuous value. To calculate gain we must sort attribute values in ascending order as follows:

{65, 70, 70, 70, 75, 78, 80, 80, 80, 85, 90, 90, 95, 96}

Then afterwards we remove the duplicates.

{65, 70, 75, 78, 80, 85, 90, 95, 96}

Gain Calculation for the attribute continuous humidity using C4.5 Algorithm

Table 3.3 Gain calculation for Continuous Values

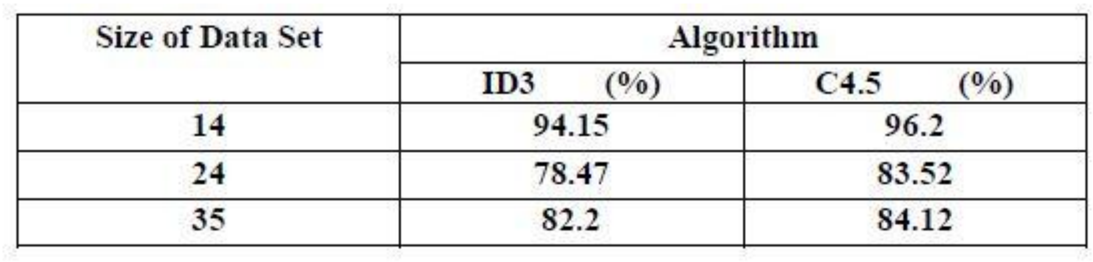
	65		70		75		78		80		85		90		95		96	
Interval	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>
Yes	1	8	3	6	4	5	5	4	7	2	7	2	8	1	8	1	9	0
No	0	5	1	4	1	4	1	4	2	3	3	2	4	1	5	0	5	0
Entropy	0	0.961	0.811	0.971	0.721	0.991	0.65	1	0.764	0.971	0.881	1	0.918	1	0.961	0	0.94	0
Info(S,T)	0.892		0.925		0.8950		0.85		0.838		0.915		0.929		0.892		0.94	
Gain	0.048		0.015		0.045		0.09		0.102		0.025		0.011		0.048		0	

Gain(S, Humidity) = 0.102

After a number of calculations, as in the ID3 example Outlook has the largest value of Information Gain hence it becomes the root of the decision tree.

According to an experiment run on the same dataset to test accuracy of the ID3 and C 4.5 Algorithm by, the C4.5 was reasonable a more accurate classifier as compared to the ID3 algorithm

Table 3.4 Accuracy Comparison between ID3 and C 4.5



Size of Data Set	Algorithm	
	ID3 (%)	C4.5 (%)
14	94.15	96.2
24	78.47	83.52
35	82.2	84.12

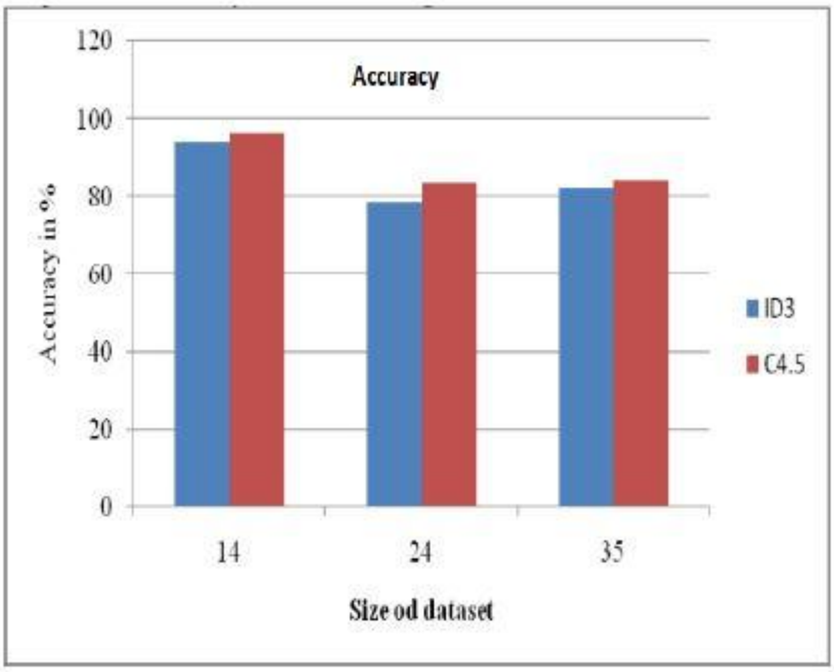


Fig 3.7 Comparison of Accuracy between ID3 and C 4.5

Although the C4.5 is more accurate than ID3 the algorithm has a slower execution time.

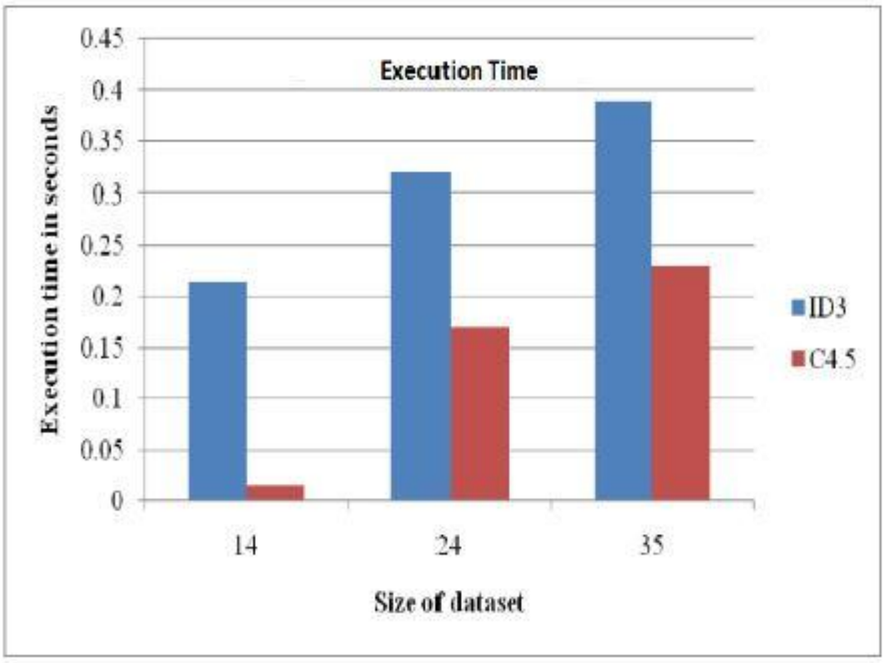


Fig 3.8 C4.5 and ID3 Execution Time

3.6 Artificial Neural Networks

Introduced in 1943 by McCulloc and Pitts, Artificial Neural Networks (ANN) are the artificial representation of the working human nervous system. They are simply an interconnected web of nodes, which are called neurons, that are connected with edges typically have a weight that adjusts as learning continues. Fig 3.9 show a representation of ANNs

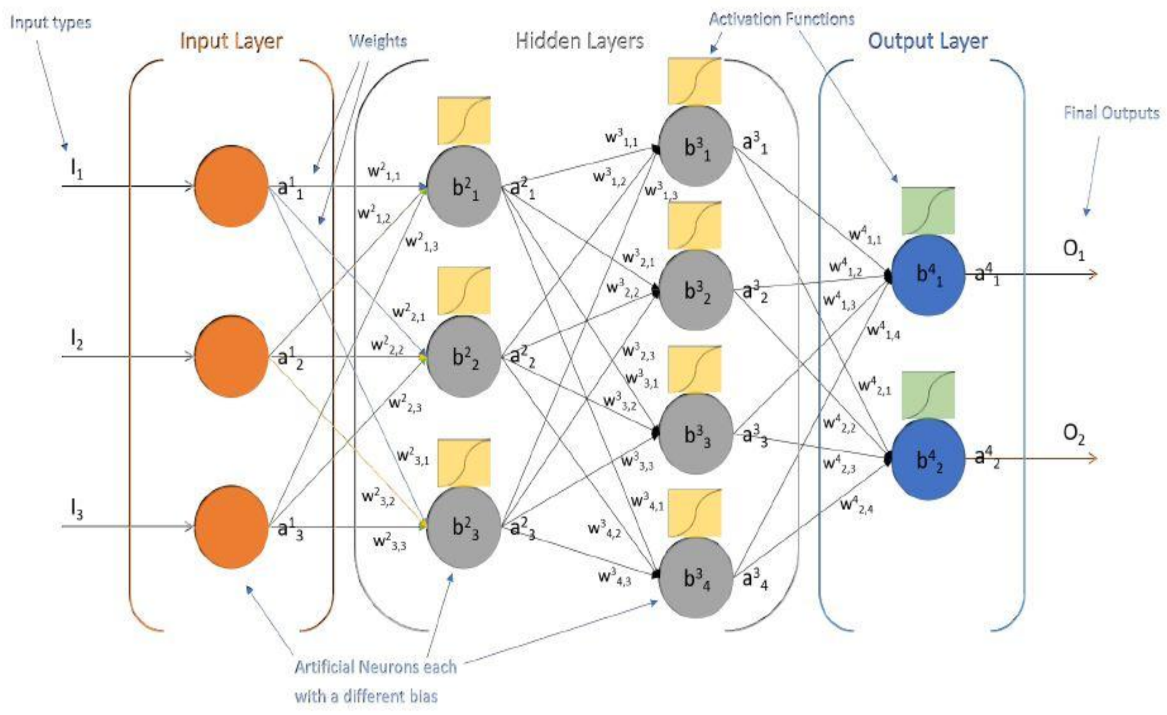


Fig 3.9 A visual representation of ANNs

Artificial Neural Networks have two learning approaches, supervised learning and unsupervised neural network learning. In Supervised learning, the neural network is provided with a labelled training set which learns a mapping of outputs y from inputs x, given a labelled set of inputs-outputs pairs.

$$\boldsymbol{d} = \{(\boldsymbol{x_i}, \boldsymbol{y_i})\}_{\boldsymbol{i=1}}^{\boldsymbol{N}} \tag{3.14}$$

Where d is the training set and N is the number of training examples. It is assumed that $\boldsymbol{y_i}$ is a categorical variable form some infinite set $y_i \in \{1 \ldots\ldots C\}$.

Multi-Layer Perceptron is an example of a supervised ANN algorithm which was used by Biven et al (2002) to detect Network Intrusion Detection.

In unsupervised Neural Network learning, the ANN has an input of $d = \{x_i\}_{x=1}^{N}$ that is a set of unlabelled data and the algorithm has to find patterns in the given data. Self-Organizing Map (S.O.M) is a type of ANN that is trained using unsupervised learning procedure to produce a low dimensional, discretized representation of the input space of training samples referred to as Maps.

3.7 Classification Performance of Machine Learning Algorithms

In Machine Learning the performance of an algorithm is measured by how they can accurately classify data to detect anomalies in data being received and processed, this is achieved by analysing data through a process called "Data Mining" which leads to the overall "Classification" of Data as being legitimate or illegitimate. 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). Classification, also known as classification learning, is a major data mining technique used to explore and find the hidden knowledge in a given 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".

Classification learning is basically a two-step process, in the first step, a classification model, mostly known as a classifier is constructed from a predetermined set of labelled 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 is represented by $\boldsymbol{n}$ attributes, $\boldsymbol{X} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots \dots \dots .. \boldsymbol{x_n})$. 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 $\boldsymbol{c} \in \{\boldsymbol{0}, \boldsymbol{1}\}$ 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 the Machine Learning classifier largely depends on how well it categories the test instances after the completion of the training phase.

3.7.1 Bias and Variance Trade-Off in Classification

There are a number of factors that impact the performance of machine learning classifiers. The concept of bias and variance is inescapable with regards to the performance of classification algorithms. When a classification model that is complex given the training dataset is inducted, this leads to over-fitting attributed to too much variance in the model. However, if a simple 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 3-8, a bias/variance compromise has to be obtained for optimal classification performance of a learner (Brain & Webb, 1999).

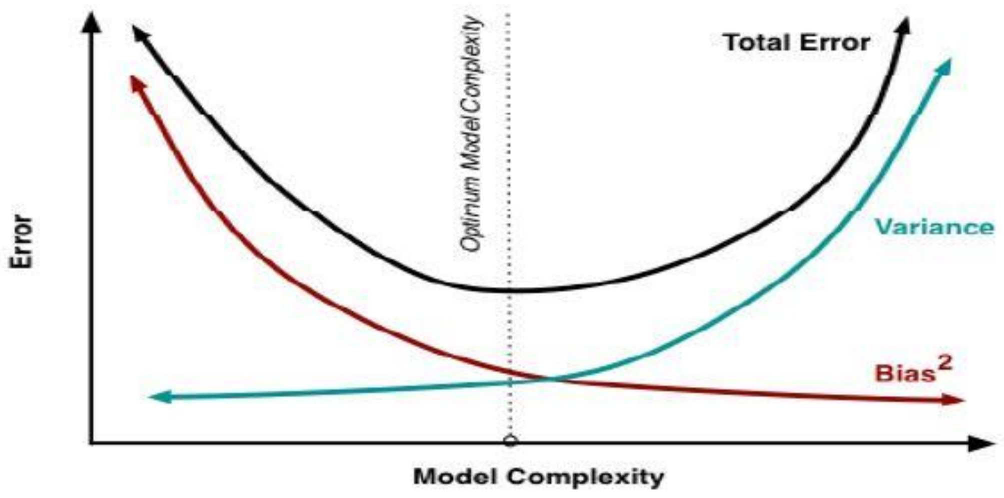


Fig 3.10 Bias-Variance Trade-Off

There are a number of factors that cause bias and variance and hence affecting the performance of classification models. In this study we investigate (1) Sample Size and (2) Type of Discretization on how they affect classification performance.

These factors, each has unique bearing of the classification performance. It is however interesting to investigate how each factor impacts classification of the proposed model, and algorithms. A number studies have been conducted to date however they are not conclusive or binding. We review accordingly the literature in the following section.

3.7.2 Effect of Training Sample Size

The size 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 influences classification performance of a classifier:

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

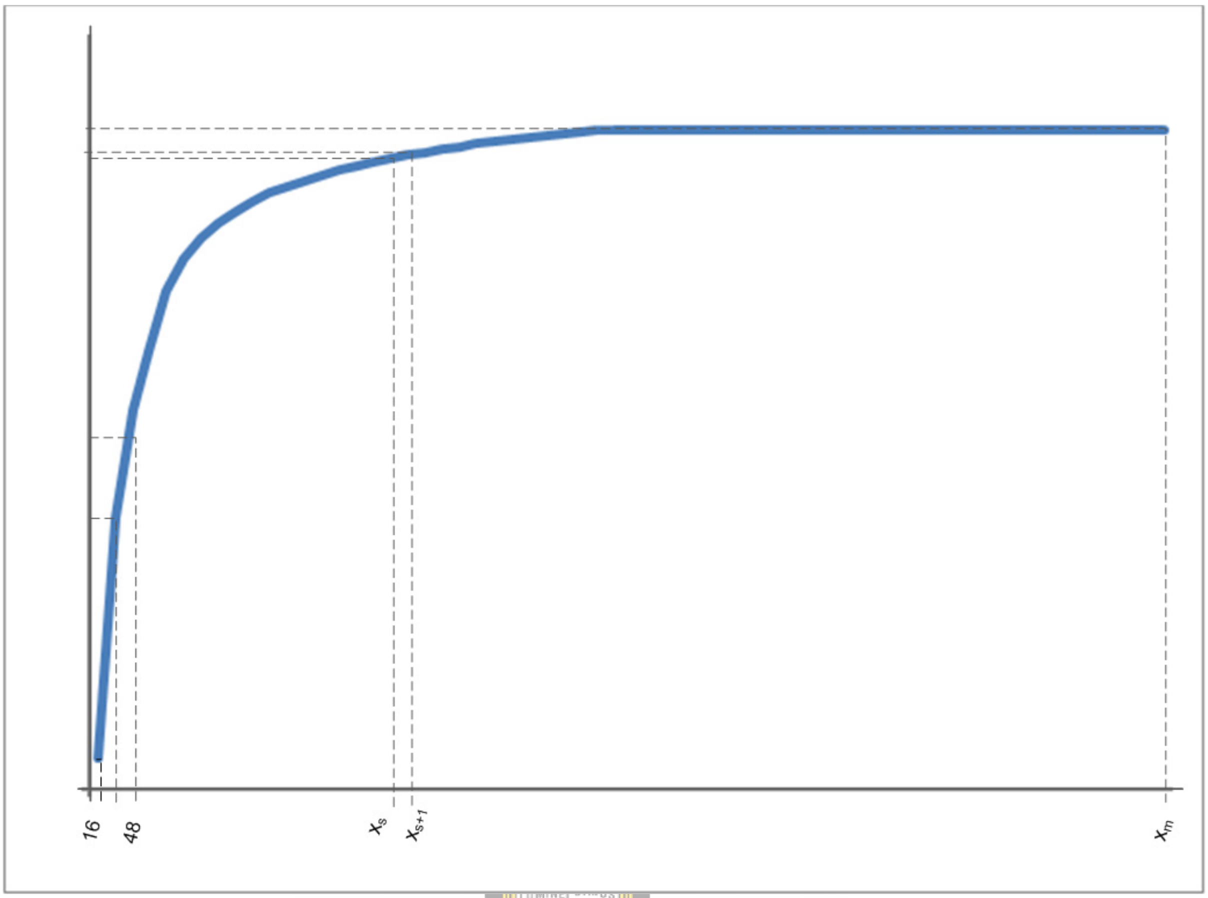


Fig 3.11 Generic Learning Curve (Source: Figueroa, 2012)

The image depicted above is an example of a generic learning curve. Beleites et al. (2012) explains a learning curve as ―*The learning curve describes the performance of a given classifier for a problem as function of the training sample size".*

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, 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). Which generally means the larger the training sample the more accurate the classifier is, and if the training set is of a small number there is a high probability of incorrectly classified data in the test set. However the number of training set is linear to time complexity, hence 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).

According to a study done by Sordo and Zeng (2005) who investigated the impact of sample size on classification accuracy of three classification algorithms SVM, Naïve Bayes and Decision Trees in classifying smoking data. Naïve Bayes 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 Naïve Bayes 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.

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:

$$\boldsymbol{H(X) = \sum_{i=1}^{n} p(x_i) I(x_i) - \sum_{i=1}^{n} p(x_i) \log_b p(x_i)} \tag{3.15}$$

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 3.15 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.

From academia and literature we note that the training sample set plays a pivotal role in the classification performance of machine learning classifiers. The relationship between the sample size and classification performance must not be overlooked. It is fundamental to determine the amount of training size that will result in optimal performance of a classifier. 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.

3.7.3 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). Most machine learning methods approach the classification task with the assumption that example attributes are discrete in nature. However this is not always the case as real life datasets that contain significant information that comes in continuous format. The process of discretization 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. The diagram below shows a representation of the discretization process.

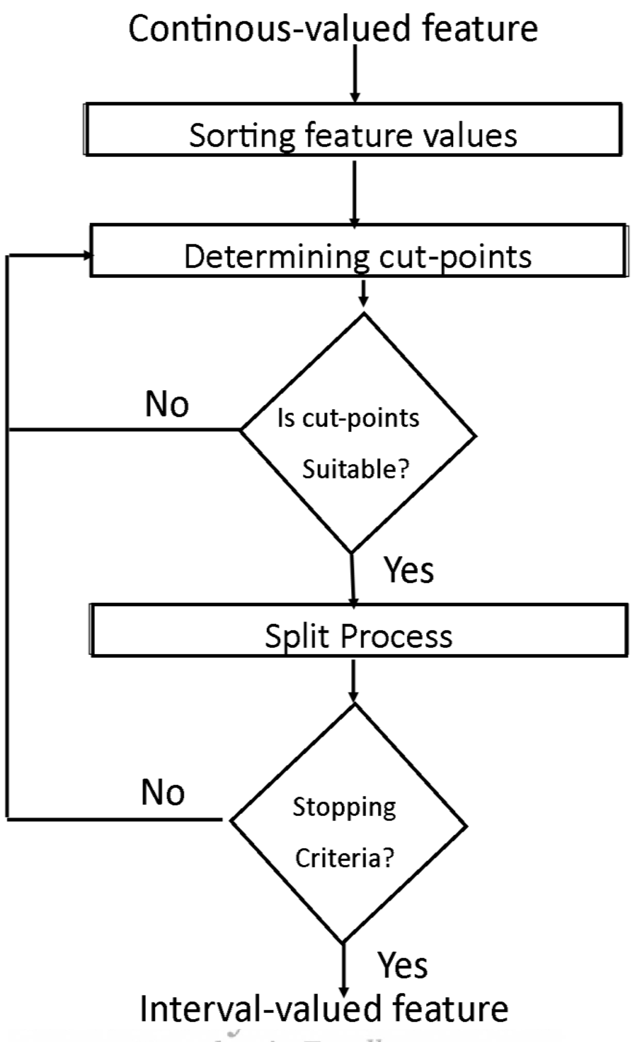


Fig 3.12 Discretization Process

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 four different axes in the classification of discretization approaches (Dougherty, Kohavi & Sahami, 1995). Discretization methods are classified as either:

1. **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.

2. **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.
3. **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.

4. **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.

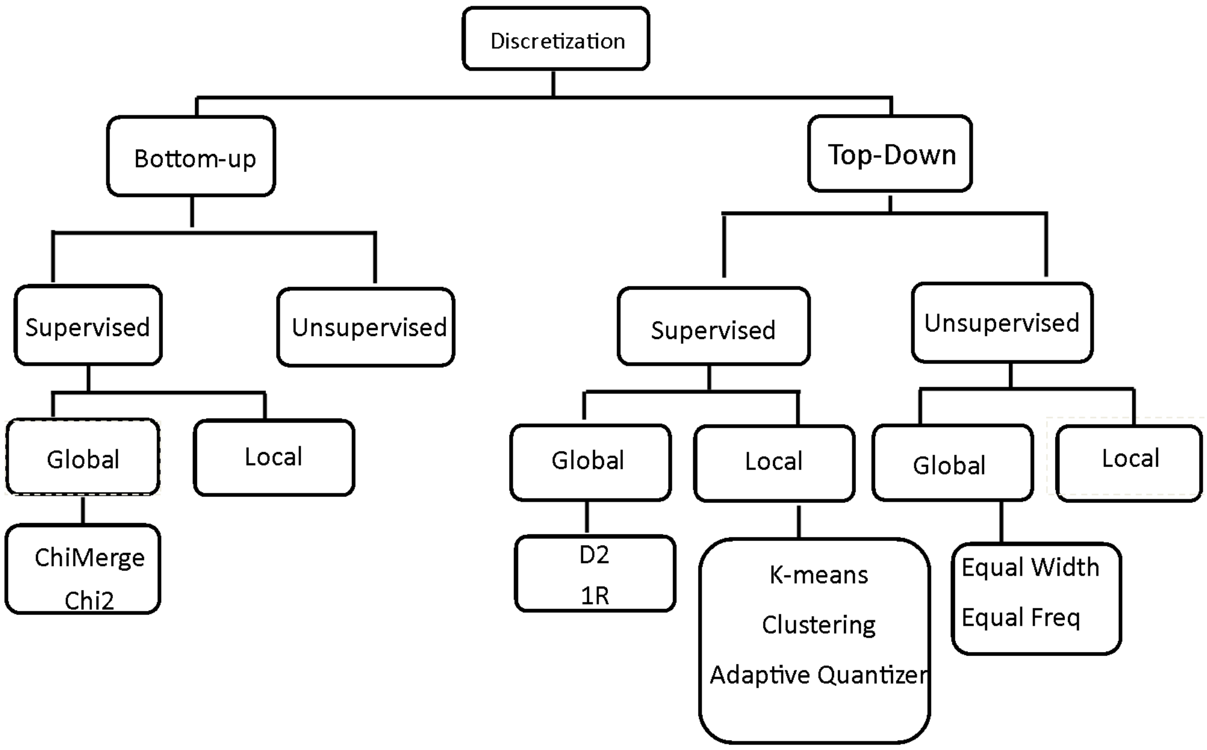


Fig 3.13 Hierarchical Representation of Discretization Methods

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).

In the following section we discuss the discretization methods used in this particular study.

Equal Width Discretization

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 $\boldsymbol{x_{min}}$ and $\boldsymbol{x_{max}}$, then this method computes the bin width by:

$$\delta = \frac{\boldsymbol{x_{max}} - \boldsymbol{x_{min}}}{k} \qquad \textbf{(3.16)}$$

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

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 $\boldsymbol{O(t)}$**,** being the number of data instances. For our empirical study, we will use a EW with k value set at 10.

Equal Frequency Discretization

This discretization algorithm sorts values in an ascending order and divides the range into $\boldsymbol{k}$ bins so that each one contains approximately the same number of training instances (Dougherty, Kohavi & Sahami, 1995). Hence, every bin contains $\boldsymbol{t/k}$ 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 $\boldsymbol{k}$ intervals with exactly the same number of values. Time complexity for this algorithm is $\boldsymbol{O(t \log t)}$ as it is necessary to perform an ordering of the data.

PKI Discretization

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.

$$\boldsymbol{p * t = N} \tag{3.17}$$

$$\boldsymbol{p = t} \tag{3.18}$$

where $\boldsymbol{N}$ represents the number of training instances, $\boldsymbol{p}$ represents the number of bins and $\boldsymbol{t}$ represents the number of training instances per partition.

The Choice of Discretization Method

The above reviewed discretization approaches are used in this study and can be used with a number of classifiers. 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 study attempts to investigate how the choice of discretization impacts the classification performance. 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 of discretization impacts classification performance, we study each of the attributes. García et al. (2013) suggests that performing empirical experiments using a set of classifier models and discretization approaches helps to identify the best performing approach. Furthermore, we examine how the training sample size impacts the discretization process and consequently the classification performance of machine learning classifiers.

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.

3.8 Related Work

According to Sherasiya and Upadhay (2016), the IoT network is similar to the Internet network in that they are exposed to the same cyber-attacks such as Man in the Middle, Denial of Service and Spam, but however mention that due to small computing resources using traditional

Internet detection and mitigation techniques to protect the IoT network would be challenging. They propose a light weight IDS using Machine Learning, their model aims at detecting nodes with multiple identities and is only valid in case of wireless networks, since it relies on wireless a wireless network to disclose genuine nodes. Although effective their model is more of a patrolling model rather than a machine learning approach and it does not detect anomaly in payload nodes.

Granjal, Silva and Lourenço (2018) propose an anomaly detection system in CoAP Sensor Networks as a solution to application layer and DoS attacks in 6LoWPAN and CoAP communication environments. The algorithm used to perform the detection of intrusions is the Support Vector Machines (SVM). SVMs are powerful classification methods that perform well using reasonable amounts of computational resources. This is an important requirement to take into account on these systems, which makes the use of Neural Networks infeasible due to their excessive computational requirements and time-consuming learning process. SVMs have a faster classification procedure, which facilitates a real time implementation, and they have the advantage of being non-linear classifiers, based on the kernel function that is used (Granjal et al, 2018). The Intrusion Detection mechanism was implemented using 2 different approaches i.e. the multi-class for detecting which intrusion had been detected and binary class to detect if there was an intrusion. For implementation they used IoT-Lab as a platform for deployment of the experiments, collecting data and performing analysis on results.

Butun et al. (2015) approach IoT anomaly detection from a different angle focusing on IoT in the cloud, which seems to be an extremely interesting approach due to the explosion of cloud computing and big data technologies. The authors analyse the current challenges of security in IoT. Firstly, the size and number of data pumped into the cloud from IoT devices is massive.

E. Hodo et al (2016) propose the use of Artificial Neural Networks as a solution to threat classification, detection and analysis. A multi-level perceptron, a type of supervised ANN, is trained using internet packet traces, then is assessed on its ability to thwart Distributed Denial of Service (DDoS/DoS) attacks. Their experiment retains 99.4% classification in accuracy. Their model demonstrates that the ANN algorithm implemented is able to successfully detect DDoS/DoS attacks against legitimate IoT network traffic.

Table 3.5 Related Work

Author(s)	Model	Accuracy
Hodo et al (2016)	Artificial Neural Network intrusion detection system for IoT attacks	99.4%
Granjal et al (2018)	Intrusion Detection and Prevention in CoAPWireless Sensor Networks using Anomaly Detection	93%
Pongle and Chavan (2015)	An IDS designed to detect wormhole attacks in IoT devices using three algorithms to detect anomalies in an IoT network.	94%
Thamilarasu and Chawla (2019)	Deep Learning Intrusion Detection Model for the Internet of Things	95%

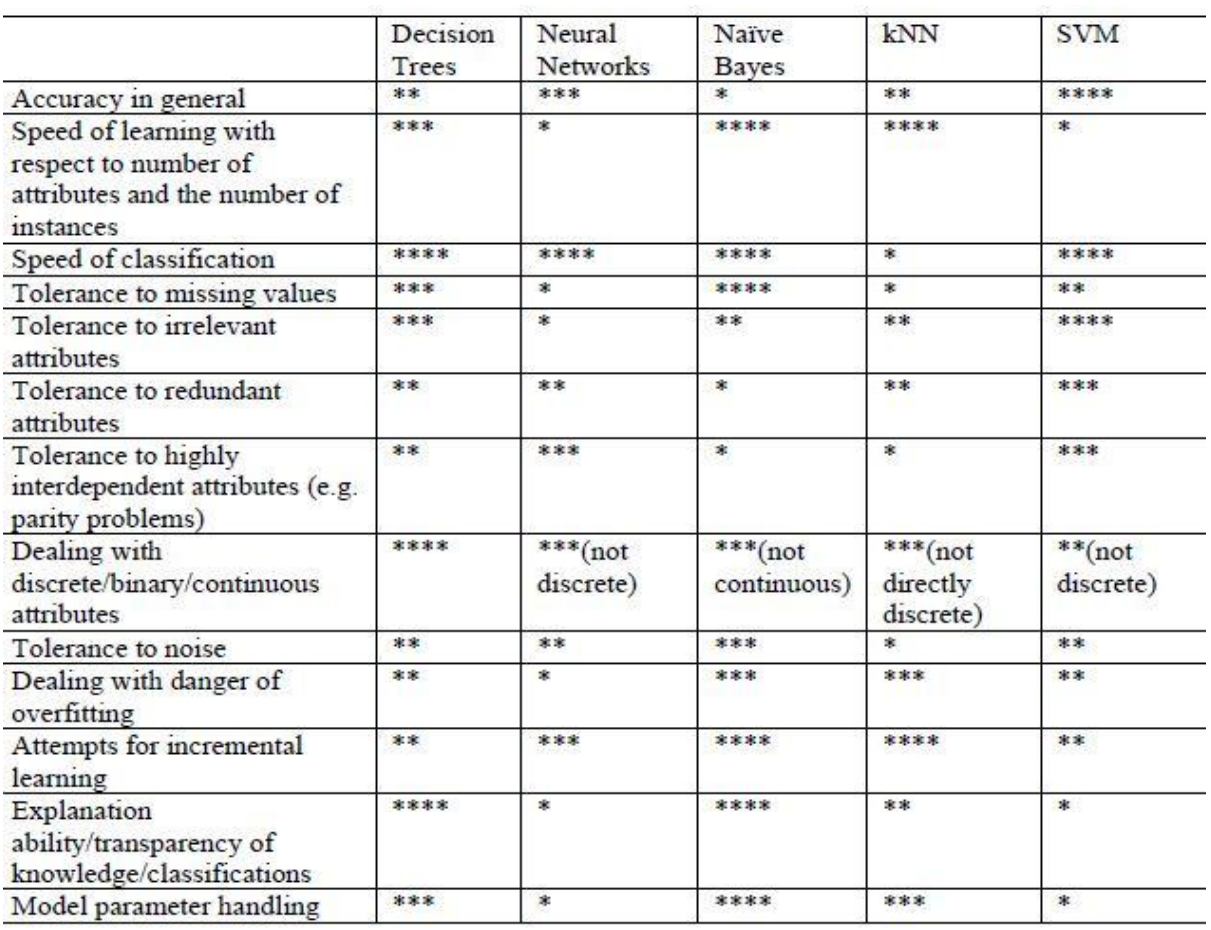
Table 3.5 represents the related work reviewed, all the authors employ various machine learning techniques to develop an intrusion detection mechanism in IoT networks with relative success. The work carried out by these researchers proves that Machine learning techniques can be employed to build a comprehensive defence mechanisms against attacks on IoT Ecosystems.

3.9 Algorithm Comparison

In Section 3.9 we compare the various algorithms discussed earlier in the chapter to critical analyse each of the algorithms strengths and weaknesses so as to determine which might be best suited for this particular study.

Table 3.6 gives a graphical comparison, of Artificial Neural Networks, Support Vector Machines, Naïve Bayes, KNN and Decision Tree Algorithms, scoring each algorithm in a particular category.

Table 3.6 A comparison of machine learning algorithms (** represents the best and * worst) (Source: Kotsiantis, 2007)**



	Decision Trees	Neural Networks	Naïve Bayes	kNN	SVM
Accuracy in general	**	***	*	**	****
Speed of learning with respect to number of attributes and the number of instances	***	*	****	****	*
Speed of classification	****	****	****	*	****
Tolerance to missing values	***	*	****	*	**
Tolerance to irrelevant attributes	***	*	**	**	****
Tolerance to redundant attributes	**	**	*	**	***
Tolerance to highly interdependent attributes (e.g. parity problems)	**	***	*	*	***
Dealing with discrete/binary/continuous attributes	****	***(not discrete)	***(not continuous)	***(not directly discrete)	**(not discrete)
Tolerance to noise	**	**	***	*	**
Dealing with danger of overfitting	**	*	***	***	**
Attempts for incremental learning	**	***	****	****	**
Explanation ability/transparency of knowledge/classifications	****	*	****	**	*
Model parameter handling	***	*	****	***	*

From table 3.6 it can be noted that the ANN and SVMs are generally more accurate and faster in-terms of speed of classification than the other algorithms in comparison, however Decision Trees and KNN can be trained at a faster rate than both sets of algorithms and have the distinct advantage of being able to handle discrete, binary and continuous datasets.

SVMs have been proven to be highly efficient in classifying large datasets, however these datasets must be linearly separable. If the dataset not linearly separable it will be transferred into a higher dimensional space so that a maximum-margin can be plotted (Yazci, Basurra & Gaber, 2018), this is not suitable for an IoT project as this process is computationally expensive and most IoT objects have minimal computational power and memory.

According to Kohavi, 1996; Domingos & Pazzani 1997, Decision Trees have been proven to outperform Naïve Bayes Algorithms in-terms of accuracy as the Naïve Bayes performs poorly with larger datasets. While the KNN algorithm is relatively easy to implement, it however does require the whole training set to be stored, which is computationally expensive in large dataset (Kongshavn Rønning, 2017), hence this may not be suitable for an IoT model.

In this study we use the C 4.5 Decision tree algorithm, which is an extension of the ID3 Decision Tree Algorithm (Mahdavinejad et al, 2015), this is because the C 4.5 Algorithm solves one particular problem that the ID3 does not, it is better equipped at handling over-fitting and can work on datasets that have missing values. Decision trees unlike other classification algorithms such as Neural Networks and SVMs can be used in dealing with discrete, binary and continuous dataset attributes hence making them unique in handling different types of datasets.

In terms of classification accuracy the C 4.5 Decision tree is not as accurate as compared to ANN and SVMs, hence the proposed model employs a clustering algorithm in the form of the K-means algorithm to assist with anomaly detection. The K-Means clustering algorithm divides data into groups, and labels each instance in the dataset. K-Means is employed to obtain clusters with the nearest mean before employing the classification algorithm (Kongshavn Rønning, 2017). Once clustering is complete the C 4.5 Algorithm is then used to classify the already clustered and labeled data. Clustering increases the efficiency and accuracy of the classification model.

3.10 Conclusion

This chapter gave a review of Machine Learning Algorithms used for classification in the Internet of Things. The underlying concepts were stated and comprehensive definitions were provided. Furthermore, the chapter reviewed related work, on anomaly detection and the challenges that other researchers faced when designing models to solve the issue of intrusion detection in IoT. The Chapter concludes by discussing the various algorithms discussed previously in the chapter and critically analysing which ones will be suitable for this particular study and justifying the choice of the algorithms chosen for the implementation of the proposed model.

Chapter 4: Methodology and Implementation

The following Chapter discusses the research and software methodology; how the proposed mechanism for anomaly detection in the Internet of things using K-Means clustering algorithm and C4.5 decision tree algorithm. It also consists of the description of the type of data that is used in implementation, building and testing this particular model, as well as the description of how the two algorithms can detect and classify anomaly in data. Furthermore, the Chapter also reviews the evaluation metrics and criteria by which the accuracy of our proposed model can be measured. The proposed anomaly detection model was trained and tested using MATLAB Math Simulator. Implementation of the proposed model consists of four stages namely input data pre-processing, clustering, training, classification and testing. This chapter also provides code-snippets of the MATLAB code used to implement the model, and brief descriptions of what the snippets do.

4.1 Methodology

Kothari (1990) states that there are basically two types of research methodologies namely quantitative and qualitative, he then points out that the former is categorized to experimental, simulation and inferential. Inferential approach entails forming a database from which to deduce the characteristics or relationships of the sample population (Silhavy et al., n.d.). Experimental involves conducting empirical studies with the aim of obtaining results from a real-world test-bed. Simulation is used for research that involves complex phenomena and hence cannot be setup in a laboratory environment (Silhavy et al., n.d.). For the purposes of this study, a simulation methodology was used, as the research involved implementing a complex IoT Interconnected ecosystem and there were no actual equipment to perform the study physically as with the experimental approach.

4.2 Model Design

The study is conducted under the framework that is shown in Fig 4.1. The purpose of the model is to apply Machine Learning Algorithms in detecting anomaly in application level data that is being sent and collected by IoT devices or sensors at the application layer of the IoT Architecture. We make use of a UCI repository IoT dataset (http://archive.ics.uci.edu/ml/machine-learning-databases/00360/AirQualityUCI.zip) which contains readings of a gas multi-sensors deployed to measure air quality in an Italian city, this dataset is used as an input that is processed systematically and predicts if data received is normal or abnormal by use of clustering and classification algorithms. The study resorted to the freely available dataset of the Italian city as the researchers couldn't find a local dataset (South Africa) that can be used for research purposes. Figure 4.1 is a representation of the proposed model.

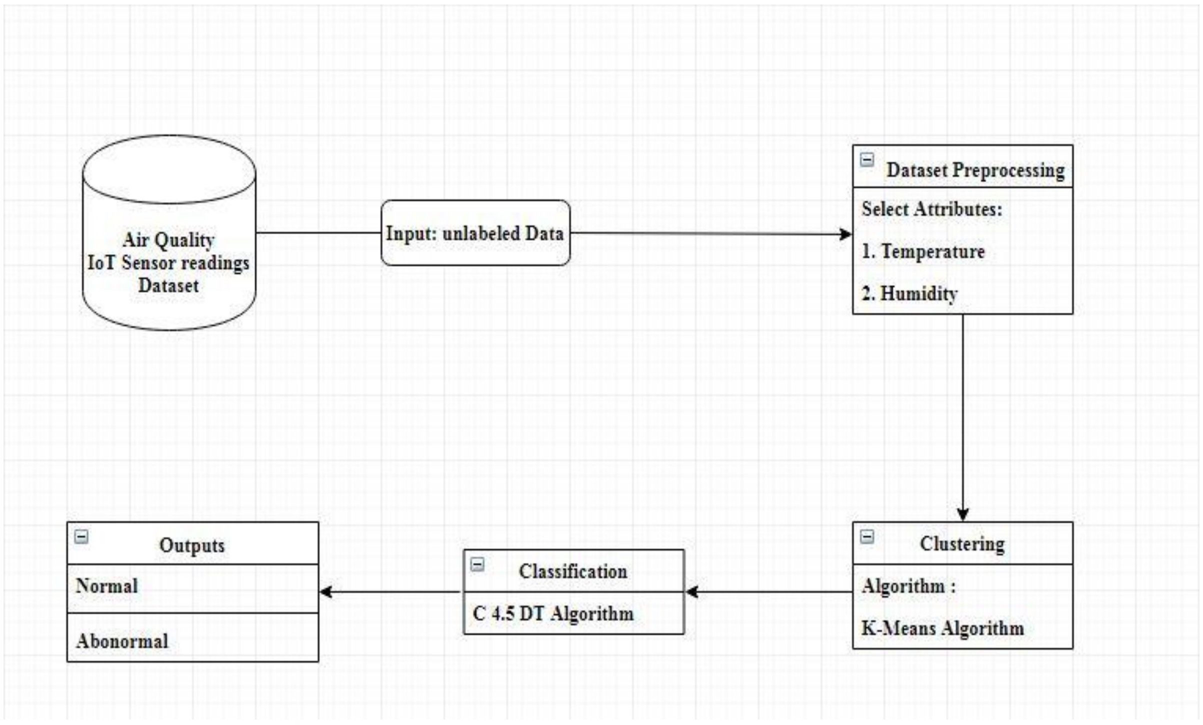


Fig 4.1 Proposed Intrusion and Anomaly Detection Model

The model follows the following steps;-

1. Training and Test dataset.
2. Perform input data pre-processing where data is processed to make it easy to process.
3. Model processing - this stage is concerned with the use of clustering and classifying algorithms for supervised learning by use of k-means algorithm for clustering and C 4.5 Decision Tree for classifying.
4. Prediction - the C 4.5 algorithm is used to predict if data received or sent is normal or abnormal.

Figure 4.2 presents the flow chart for the anomaly detection model being proposed.

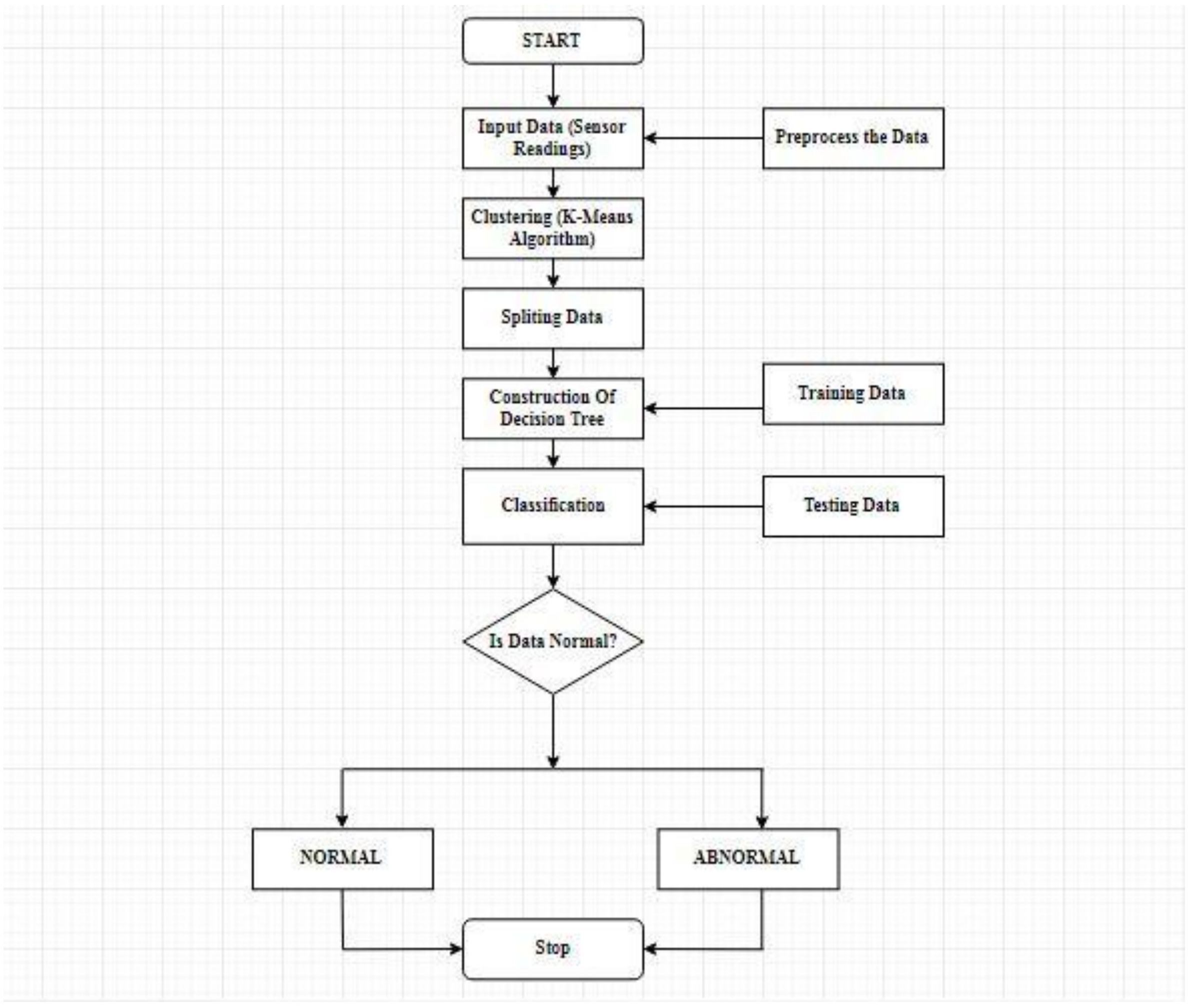


Fig 4.2 Proposed anomaly detection model flow chart

According to the flow chart depicted in Fig 4.2, the model starts by taking input data from sensor recordings, that data is then pre-processed before being passed on to the clustering algorithm. Data is then split into training and testing data, the training data is used to construct a decision tree using the C 4.5 algorithm. After the decision tree is constructed, the testing data was used to test the tree which classified data into normal or abnormal.

4.3 Input Data

To obtain a labelled dataset for intrusion detection and anomaly detection in IoT is not easy, as there are not many. There are a number of non-IoT datasets available such as the KDD Cup 1999 dataset. For this study we used a dataset acquired from UCI IoT repository that contains the readings from multiple gas sensor devices deployed to measure the quality of air in an

Italian city. The dataset has 9358 instances of timed responses from chemical sensors embedded in an Air Quality Multi-sensor Device in an Italian city. According to the explanation notes of the dataset, data was recorded for a period of a year with each sensor readings collected hourly. The sensors collected concentrations of the following gases Hydrogen Carbons, CO, Benzene, Nitrogen Oxide (NOx) and Nitrogen Dioxide (NO2) (De Vito et al.,2008).

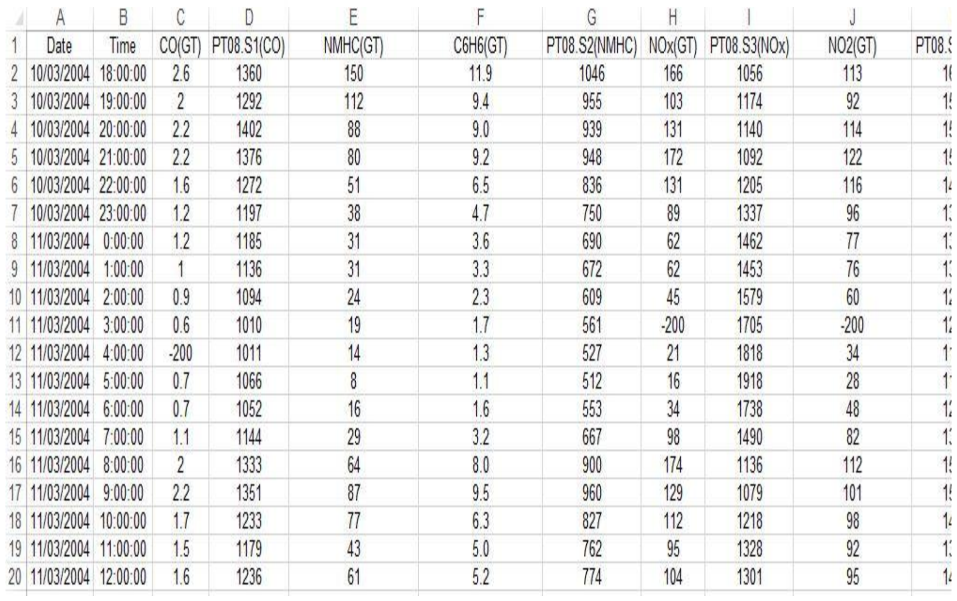
The dataset is freely available, but requires pre-processing before it can be used.

4.3.1 Data Pre-processing

The dataset obtained contains over a million readings in a Microsoft csv file, which is computationally expensive for the sensors in the IoT network. The dataset contains unnecessary attribute data such as date, time and sensor id, these were removed in the pre-processing stage and for the purposes of this study, only temperature and humidity sensor readings were used. Taking such an approach is advantageous as;

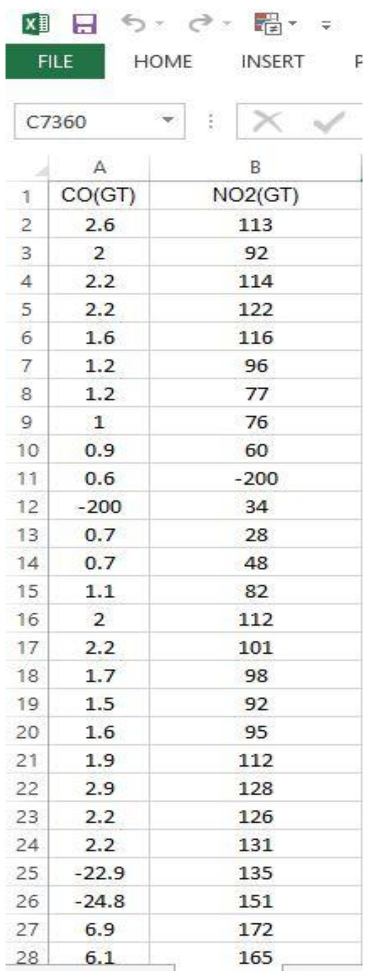
1. The data will become labelled and known to the trainer or teacher of the model,
2. This overcomes the problem of obtaining a labelled dataset for the Internet of Things at a pricey cost.

The resulting dataset that was used for training consisted of two attributes, namely Carbon Monoxide concentration and Nitrogen Oxide levels.



	A	B	C	D	E	F	G	H	I	J	
1	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S
2	10/03/2004	18:00:00	2.6	1360	150	11.9	1046	166	1056	113	1
3	10/03/2004	19:00:00	2	1292	112	9.4	955	103	1174	92	1
4	10/03/2004	20:00:00	2.2	1402	88	9.0	939	131	1140	114	1
5	10/03/2004	21:00:00	2.2	1376	80	9.2	948	172	1092	122	1
6	10/03/2004	22:00:00	1.6	1272	51	6.5	836	131	1205	116	1
7	10/03/2004	23:00:00	1.2	1197	38	4.7	750	89	1337	96	1
8	11/03/2004	0:00:00	1.2	1185	31	3.6	690	62	1462	77	1
9	11/03/2004	1:00:00	1	1136	31	3.3	672	62	1453	76	1
10	11/03/2004	2:00:00	0.9	1094	24	2.3	609	45	1579	60	1
11	11/03/2004	3:00:00	0.6	1010	19	1.7	561	-200	1705	-200	1
12	11/03/2004	4:00:00	-200	1011	14	1.3	527	21	1818	34	1
13	11/03/2004	5:00:00	0.7	1066	8	1.1	512	16	1918	28	1
14	11/03/2004	6:00:00	0.7	1052	16	1.6	553	34	1738	48	1
15	11/03/2004	7:00:00	1.1	1144	29	3.2	667	98	1490	82	1
16	11/03/2004	8:00:00	2	1333	64	8.0	900	174	1136	112	1
17	11/03/2004	9:00:00	2.2	1351	87	9.5	960	129	1079	101	1
18	11/03/2004	10:00:00	1.7	1233	77	6.3	827	112	1218	98	1
19	11/03/2004	11:00:00	1.5	1179	43	5.0	762	95	1328	92	1
20	11/03/2004	12:00:00	1.6	1236	61	5.2	774	104	1301	95	1

Input data before pre-processing



	A	B
1	CO(GT)	NO2(GT)
2	2.6	113
3	2	92
4	2.2	114
5	2.2	122
6	1.6	116
7	1.2	96
8	1.2	77
9	1	76
10	0.9	60
11	0.6	-200
12	-200	34
13	0.7	28
14	0.7	48
15	1.1	82
16	2	112
17	2.2	101
18	1.7	98
19	1.5	92
20	1.6	95
21	1.9	112
22	2.9	128
23	2.2	126
24	2.2	131
25	-22.9	135
26	-24.8	151
27	6.9	172
28	6.1	165

Fig 4.3 Dataset before and after initial data pre-processing

4.4 Clustering

Clustering is the splitting of a given dataset into a number of groups in such a way that the elements in each resulting group are of the similar attributes and are similar to each other, these particular groups are known as Clusters (Aggarwal and Reddy, 2013).

There are a number of clustering methods; centroid-based, distribution based, density based clustering. Centroid-based clustering is mostly applied to Intrusion Detection and represents each cluster of data by a central vector called a “centroid”. For the purpose of this study, K-Means Clustering algorithm was used.

In K-means, an unlabelled dataset is divided into *k* “clusters”, where data points belonging to the same cluster must have similarities, the distance between data points is used as a measurement for level of similarity. The clustering algorithm seeks to find a set of *k* cluster centres, such that the distances between data points and their nearest centre are minimized.

The input data in this study clustered groups into 2. K-means clustering technique randomly picks two records from input data and then starts to fetch other records in the same dataset and compare them with any two randomly selected records by calculating the distance between them (Aggarwal and Reddy, 2013).

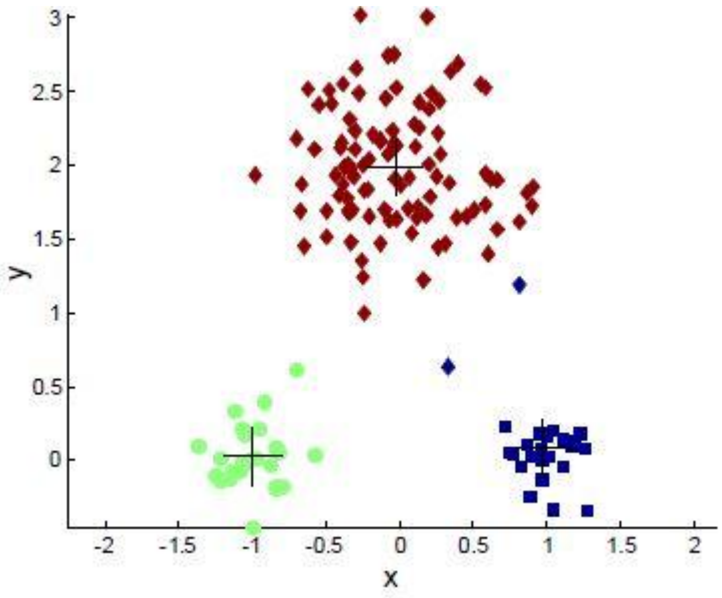


Fig 4.4 K-Means Clustering Illustration with k=3

Algorithm 1 : Basic K-means Algorithm
1. Select *K* points as the initial centroids.
2. Repeat
3. Form *K* Clusters by assigning all points to the closest centroid
4. Re-compute the centroid of each cluster.
5. **Until** The centroids don't change

Using the K-means algorithm the centroids can be found using the following formulas (Aggarwal & Reddy, 2013):

$$\boldsymbol{m_k} = \frac{\sum_n \boldsymbol{r_{kn} x_n}}{\sum_{j,n} \boldsymbol{r_{jn}}} \qquad (4.1)$$

Where
$$\boldsymbol{r_{kn}} = \frac{\mathbf{exp}\left(-\boldsymbol{\beta d}(\boldsymbol{x_n},\ \boldsymbol{m_k})\right)}{\sum_j \mathbf{exp}\left(-\boldsymbol{\beta d}(\boldsymbol{x_n}, \boldsymbol{m_j})\right)} \tag{4.2}$$

$d(a, b)$ is the Euclidean distance between a and b.

4.5 Classification

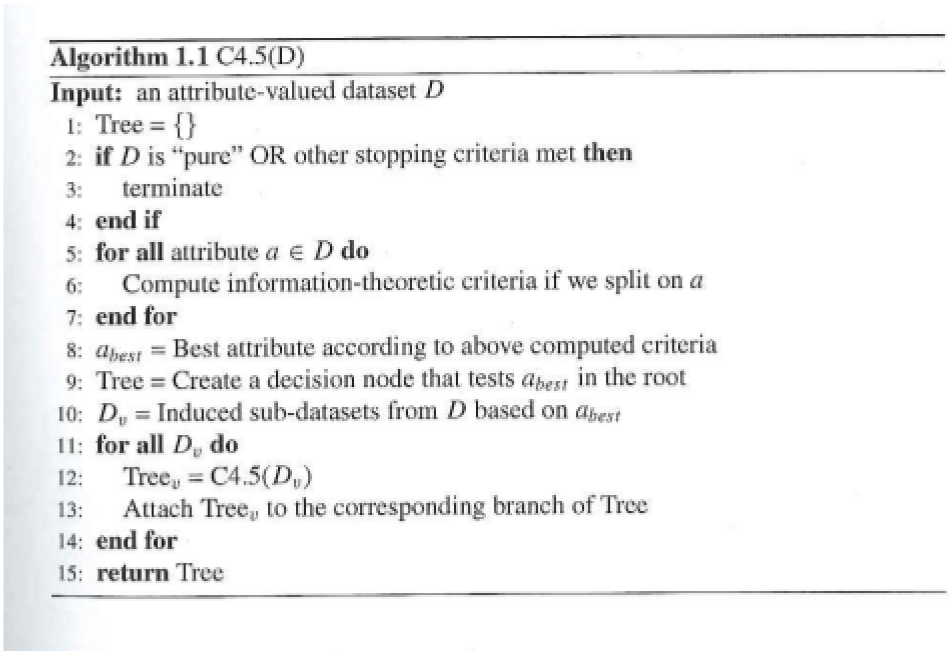
Decision Tree (DT) Algorithms is one of the most widely used algorithms in classification (Quinlan, 2014). A DT is built in two phases, namely a growth phase and a prune stage. The construction of the preliminary tree is called the ‘growth phase’. After building the preliminary tree, a sub tree is then constructed, this process is called pruning. The tree structure is made up of nodes and branches, each leaf of the tree represents a class, and in this case there are 2 classes, that is normal or abnormal. The classes are extracted as a result from data clustering using K-means algorithm, each branch represents a conjunction leading to a class label (Quinlan, 2014).

For the purpose of this study, the chosen decision tree algorithm is the C 4.5 algorithm. The C 4.5 algorithm is used to generate the decision tree. The algorithm was first proposed by Ross Quinlan (1986). The C 4.5 algorithm is given a set of training data of already classified sample. The input in the clustering process in split into two groups;

- Training set – this will contain records that have been classified to normal and abnormal
- Testing set – this will contain records that will be used for testing the model built in training.

With the training set used as an input, the C 4.5 constructs the tree. Each node on the tree consists of a temperature and humidity value. The algorithm builds the tree leaves by use of class attributes: normal and abnormal, this happens through the use of if statements (Quinlan, 2014).

The basic pseudo code for C 4.5 algorithm is as follows (**Nor** .S., et al, 2017):



```
Algorithm 1.1 C4.5(D)
Input: an attribute-valued dataset $D$
Tree = {}
if $D$ is "pure" OR other stopping criteria met then
    terminate
end if
for all attribute $a \in D$ do
    Compute information-theoretic criteria if we split on $a$
end for
$a_{best}$ = Best attribute according to above computed criteria
Tree = Create a decision node that tests $a_{best}$ in the root
$D_v$ = Induced sub-datasets from $D$ based on $a_{best}$
for all $D_v$ do
    Tree$_v$ = C4.5($D_v$)
    Attach Tree$_v$ to the corresponding branch of Tree
end for
return Tree
```

Fig 4.5 C 4.5 Algorithm

4.6 Implementation

For the implementation and testing of the proposed Intrusion/Anomaly Detection Model, we use MATLAB (see 4.6.1 for description). In our MATLAB program we implement a number of functions stated in table 4.1:

Table 4.1 MATLAB functions used in Intrusion Detection Model

Program/Function	Description
Clustering.m	This file implements Clustering, as the main program, it accepts input data from a Microsoft excel file, converts that particular data into a MATLAB matrix, the program calls the KM.m file which contains the function that implements the clustering, clustering the input data to form centroids, and then saves the clustered data in a

	Microsoft excel file, a csv file that MATLAB can process.
KM.m	This file contains the function that actually implements the K-Means Algorithm. The function in the file accepts input data in form of a matrix and derives centroids based on the value of K clusters, then it returns the result.
Classification.m	This MATLAB file accepts the results from Clustering.m in the form of a matrix and splits the data into two parts, one will be the training set the other the testing set. The file calls the C 4.5 Algorithm which will use the data to produce a decision tree.
C4-5.m	This file implements the C4.5 Decision tree algorithm. It accepts a matrix consisting of the training set, which has cluster ids identifying which centroid each row belongs to. The file calls the build_tree.m file to construct the decision tree.
Build_tree.m	This is the file that is called to implement the construction of the tree, constructing the decision tree based on information-theoretic criteria.
Use_dt.m	This MATLAB uses the decision tree that is built by the C 4.5 algorithm, and applies the testing matrix. The data compares the results of classification with the input cluster ID from the testing dataset. The function then produces results for True Negative, True Positive, False Positive and False Negative.

The code files presented above are available in Appendix A.

4.6.1 Tools chosen

This study proposed a classification model that uses K-Means Clustering and C 4.5 Decision Tree Algorithm as an intrusion detection mechanism that classifies data as normal or abnormal. To determine how the model measures against other classification algorithms, a few algorithms that have been used in intrusion detection models where chosen. These algorithms were chosen according to their well-known performance as found in a number of literature sources as discussed in Chapter 3. The algorithms are: the Naïve Bayes Classifier and Multi-Layer Perceptron Artificial Neural Network. The proposed Model that uses K-Means and C 4.5 Algorithm is implemented in MATLAB Math Simulator and both the Naïve Bayes and MLP ANN are implemented in WEKA due to its vast functions that allow implementation of both algorithms.

MATLAB

MATLAB is a high level performance language that is used for mathematics, statistics and technical computing. It manages to integrate computation, simulations, visualization, and programming in a Graphic User Interface environment where problems and solutions are expressed in familiar mathematical notation. The basic data element of MATLAB is the matrix which is an array that does not require dimensioning, a simple integer value is considered as a matrix of one column and one row.

WEKA

Waikato Environment for Knowledge Analysis (WEKA) was used in this study to implement the Naïve Bayes algorithm and Multi-Layer Perceptron Artificial Neural Network, this is a tool specifically made for machine learning purposes by the University of Waikato in New Zealand. WEKA Tool has a collection of Machine Learning algorithms and data pre-processing tools that can be used by researchers and practitioners (Hall et al., 2009). This includes algorithms for clustering, classification and regression. The graphical user interfaces and visualization options in WEKA can be used for data exploration and algorithm evaluation. WEKA supports data pre-processing in different file formats such as ARFF (which is the native file format of WEKA), Matlab, CSV, ASCII files and many more (Bouckaert et al., 2010).

Other tools

During the implementation process, a number of different data mining tools were considered but some did not have the proper evaluation methods for the particular algorithms in question. IBM SPSS Statistics was used in evaluation and testing for accuracy by use of Friedman's ANOVA Test. Some of the notable tools that were not used are R and Orange. R although a powerful, data mining and statistical analysis tool, it was not used in this study as R required a rather steep learning curve. Orange was not used in the study as it does not support widgets for statistical testing and reporting capabilities are limited to exporting visual data models (Rangra et al, 2016).

The mentioned tools are run on a Windows 7 Workstation with 4GB of RAM

4.6.2 Input Data Pre-processing

As stated earlier we used an air quality dataset from the UCI Machine Learning Repository. To process the data, we used only the Temperature and Humidity recordings for our training and testing set, hence some of the measured attributes such as the gas concentrations and date and time were be removed from the dataset.

After removal of the unwanted data from the dataset, the new dataset was used as input data as shown in Fig 4.1, with attributes of Temperature and Humidity only.

4.6.3 K Means Algorithm

K-Means algorithm was implemented through the KM.m file in our MATLAB directory. The function accepts 2 inputs:

1. Input data or dataset for clustering.
2. Number of groups or clusters i.e K value

The function then returns matrix of clustered data using:

Function[M] = KM(x,K)

The function is called in the main program Clustering.m. The program Clustering.m initially imports data from our dataset which is a Microsoft Excel file on the MATLAB directory.

inputdata = “AirQualityDataset.csv’

MATLAB processes the data in the excel file and puts it in a matrix where data is stored in MATLAB’s memory for clustering.

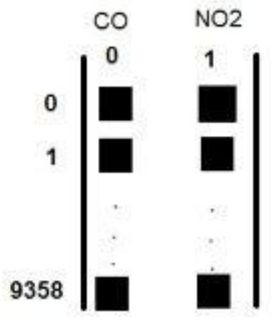


Fig 4.6 Input data representation as a MATLAB Matrix

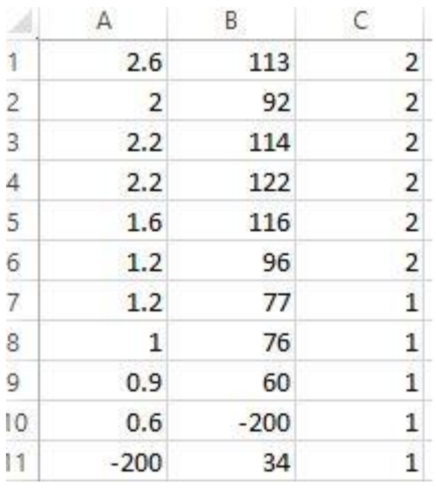
The data matrix is mapped to the KM function by use of

C = KM(M, K)

M is the matrix and K=2 i.e. the number of clusters that the algorithm had to create. The records in the input data were clustered and each record received a clustering id to determine which group or cluster that particular record belonged to. The results of the clustered data were then recorded to an external excel file using csvwrite.

csvwrite(results,R)

An example of the resulting excel file and matrix representation are shown below:



	A	B	C
1	2.6	113	2
2	2	92	2
3	2.2	114	2
4	2.2	122	2
5	1.6	116	2
6	1.2	96	2
7	1.2	77	1
8	1	76	1
9	0.9	60	1
10	0.6	-200	1
11	-200	34	1

Fig 4.7 Clustered Data saved in Microsoft Excel Sheet.

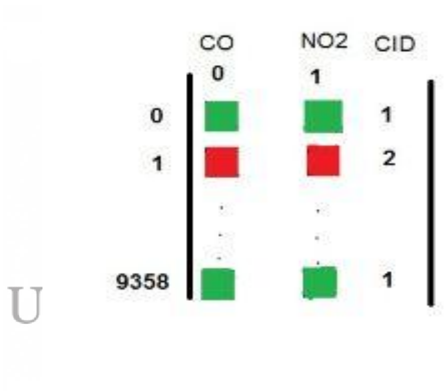


Fig 4.8 Matrix Representation after clustering.

4.6.4 C4.5 Algorithm

C4.5 DT classifies data based on probabilities. Classification.m contains the program that was used to call the C4.5 Algorithm, using the C4.5 function which implemented the classification. Before constructing the tree, data from clustering was divided into 2 parts. One became the training data and the other the testing data. The program first read data from the clustered data csv by use of

DATA = ClusteredData.csv

CDATA = csvread(DATA)

Classification divided data into 2 matrices:

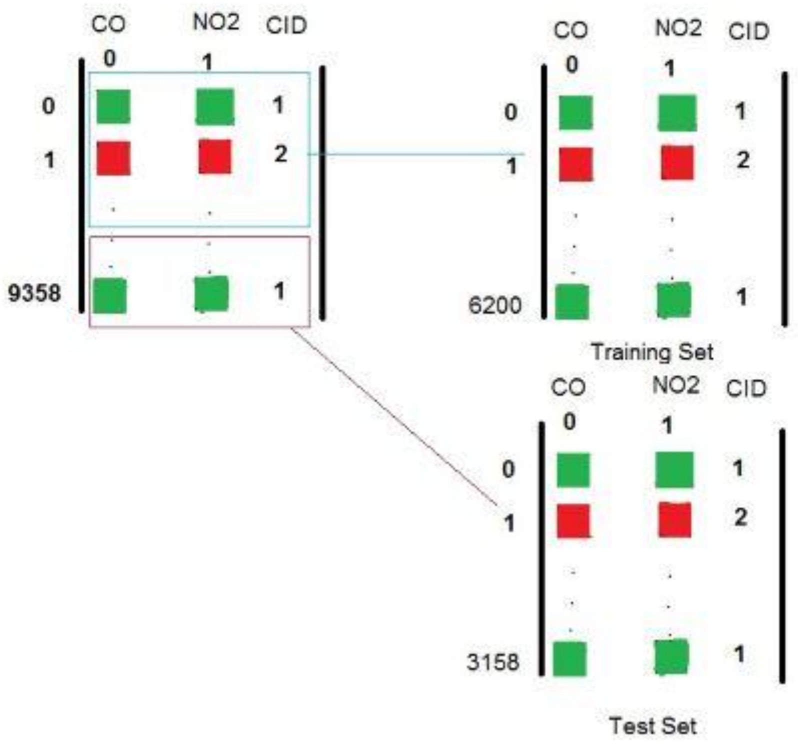


Fig 4.8 Matrix representation during classification.

4.7 Model Evaluation

There are a number of evaluation metrics that can be used to determine the performance of classification algorithms and models. They can be based on accuracy, speed, robustness and scalability. In this study, the classification performance of the Anomaly Detection classifier under study was examined during the testing stage, where the models were tested on our dataset. The classification ability was measured by its predictive accuracy. Binary classification was performed and the speed of the classifier was measured. A confusion matrix was the most appropriate way to present the binary classification results. It is a technique that is used to describe and evaluate the performance of the model (Kurniawan, Rosmansyah, & Dabarsyah, 2015). In Binary classifications, there are four possible outcomes;

- True Positive (TP) – in this scenario the classifier predicts that there are anomalies in given data and that particular data is actually abnormal, then the attempt is said to be true positive (TP). The model would have correctly classified an intrusion as intrusive.
- True Negative (TN) – here the classifier would have predicted correctly that there are no anomalies in the input data, then the prediction is a true negative (TN). The model would have correctly classified a normal event as normal.
- False Positive (FP) – here the classifier wrongly predicts that there is an anomaly in data while the data is actually normal, then the attempt is said to be false positive (FP). The model would have incorrectly classified a normal event as intrusive. The purpose of most anomaly detection models is to reduce the false positives to a minimum.
- False Negative (FN) - The model would have incorrectly classified intrusive behaviour as normal.

Table 4.2 Confusion Matrix for Classification

Actual	Classification	
	Abnormal	Normal
Abnormal	True Positive (TP)	False Negative (FN)
Normal	False Positive (FP)	True Negative(TN)

Table 4.2 is a graphic format to depict the confusion matrix. The classification problem is solved by further processing information extracted from the confusion matrix. The accuracy of the model can be calculated by the following formula (Ashour, W. and Fyfe, C. ,2007):

$$\boldsymbol{Accuracy = \frac{TP + TN}{TP + FP + TN + FN}} \tag{4.3}$$

Accuracy is the measure of the total number of events that are correctly classified including normal and intrusive events.

The misclassification rate of the proposed model i.e. how often the model or classifier got it wrong can be calculated as follows (Ashour & Fyfe, 2007):

$$\boldsymbol{Misclassification} = \frac{\boldsymbol{FP + FN}}{\boldsymbol{TP + FP + TN + FN}} \tag{4.4}$$

The recall rate or True Positive (Ashour & Fyfe, 2007):

$$\boldsymbol{TP\ Rate} = \frac{\boldsymbol{TP}}{\boldsymbol{number\ of\ actual\ abnormal\ records\ in\ (TP + FP + TN + FN)}} \tag{4.5}$$

The false positive rate can be calculated by use of the following formula (Ashour & Fyfe, 2007):

$$\boldsymbol{FP\ Rate} = \frac{\boldsymbol{FP}}{\boldsymbol{number\ of\ normal\ records\ in\ (TP + FP + TN + FN)}} \tag{4.6}$$

Specificity or TN Rate of the model can be calculated as follows (Ashour & Fyfe, 2007):

$$\boldsymbol{TN} = \frac{\boldsymbol{TN}}{\boldsymbol{number\ of\ actual\ normal\ records\ in(TP + FP + TN + FN)}} \tag{4.7}$$

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{number\ of\ predicted\ abnormal\ records\ in(TP + FP + TN + FN)}} \tag{4.8}$$

The prevalence of the classifier is calculated using the below formula (Ashour & Fyfe, 2007):

$$\frac{\boldsymbol{actual\ number\ of\ abnormal\ records\ in\ (TP + FP + TN + FN)}}{\boldsymbol{TP + FP + TN + FN}} \tag{4.9}$$

The above equations were used to evaluate the proposed intrusion detection model and compare the results.

4.7.1 Accuracy Estimation Methodologies

Accuracy is the most widely used measure for performance evaluation of classification models. The estimation methodologies that are were used in this study were based on accuracy metrics such as hit rates, error rates, etc. Estimating accuracy is important as it is necessary to verify if a model is reliable for future predictions and when more than one model or classification algorithm is involved, there is a need to have some kind of measurement or metric that can be used to choose the most efficient and accurate model among a given number (Olson, 2008). The study, made use of three methodologies, the Holdout, K-fold cross validation and ROC Curves. Brief descriptions of those methodologies are given below:-

- **Holdout** - The Holdout method involves splitting the training and test sample. This methodology is one of the simplest and most commonly used practice among the evaluation methodologies. Data is randomly split into two subsets, i.e. training and testing. According to Olson et al (2008), the most common split ratio that is used is generally selecting the training set from two-thirds of the data and testing data is the remaining third. After the data is divided into training and testing, a classification model is built through inducing, using the training data. Later on, this model is used to calculate the performance of the model constructed. This technique is used when there is sufficient data to split between training and testing set.

- **K-Fold Cross Validation** – This method is used when the amount of available data or dataset is small or limited and it is too risky to divide the data into two subsets, hence there is a need to consider other methods. A method that can be used is the k-fold cross validation or rotation estimation. In cross validation, a fixed given number "*k"* is chosen as the number of folds that are to be used. The dataset is divided into *k* mutually exclusive subsets of almost the same size. Given that we split data *D* into *k* subsets $\{D_1, _2,\ldots,D_k\}$, each of these subsets is said to be a fold . The procedure of k-fold cross validation is as follows; each and every value of D_x is used as the testing set while the rest become part of the training set, this is repeated and iterated fold by fold till all folds have been used as a testing set for at-least once. After all the iterations are done and tested, accuracy rates are calculated and added, then divided by total number of folds to find the classification rate.
- **Receiver Operating Curves (ROC)** – these are based on the confusion matrix often used for visualising classifier performance.

4.8 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 and testing set. MATLAB, our simulation software set up was introduced and algorithms for anomaly detection were discussed and implemented. This chapter also stated the performance evaluation criteria used to assess the performance of the proposed intrusion detection model and classifier. The following Chapter presents the findings and results of the experiments done and their respective discussions and findings.

Chapter 5: Results and Analysis

This chapter presents and discusses the results from the implementation described in Chapter four. The chapter goes through an evaluation of the proposed anomaly Intrusion detection model in the IoT environment. The chapter contains a number of calculated performance metrics for the proposed model from the results obtained in chapter four. The performance metrics are the values that can be calculated using the number of correct and incorrect classifications predicted by the model, these include overall accuracy, precision, misclassification rate, sensitivity, specificity and prevalence. This chapter further explores how the classifier performance and accuracy are affected by the choice of evaluation methodology, the sample size and the type of discretization approach. We further perform tests such as the Friedman's ANOVA and Wilcoxon test on statistical data gathered from particular experiments in this chapter to have sound conclusions.

5.1 Results

The model as described in chapter four, was implemented in MATLAB and the experiment began with clustering the input data. The results after clustering were recorded in a csv file and a graphical presentation is shown in Fig 5.1 and 5.2 which show input before and after clustering

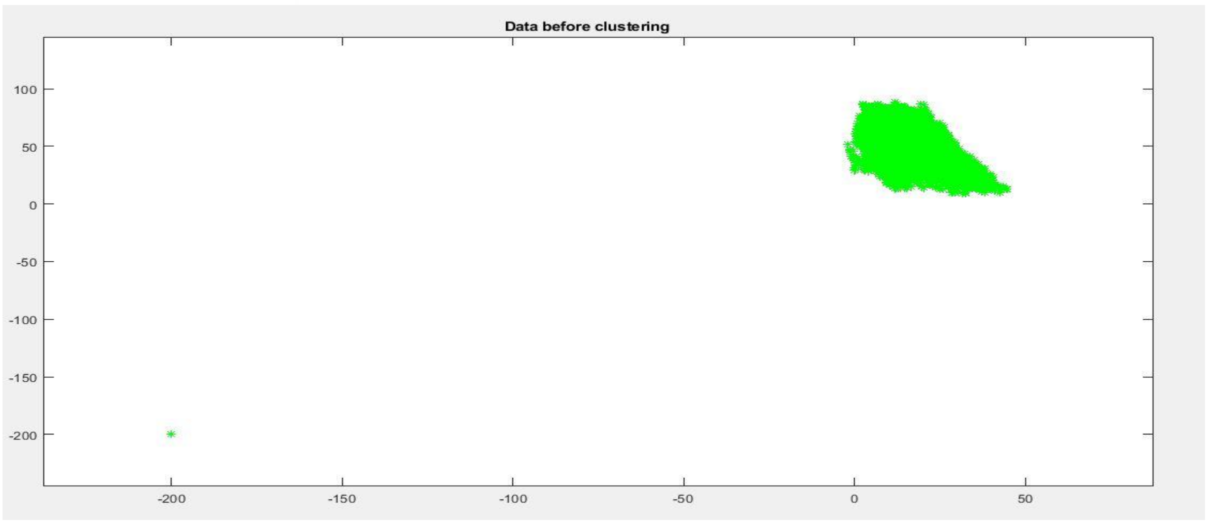


Fig 5.1 Input data before clustering

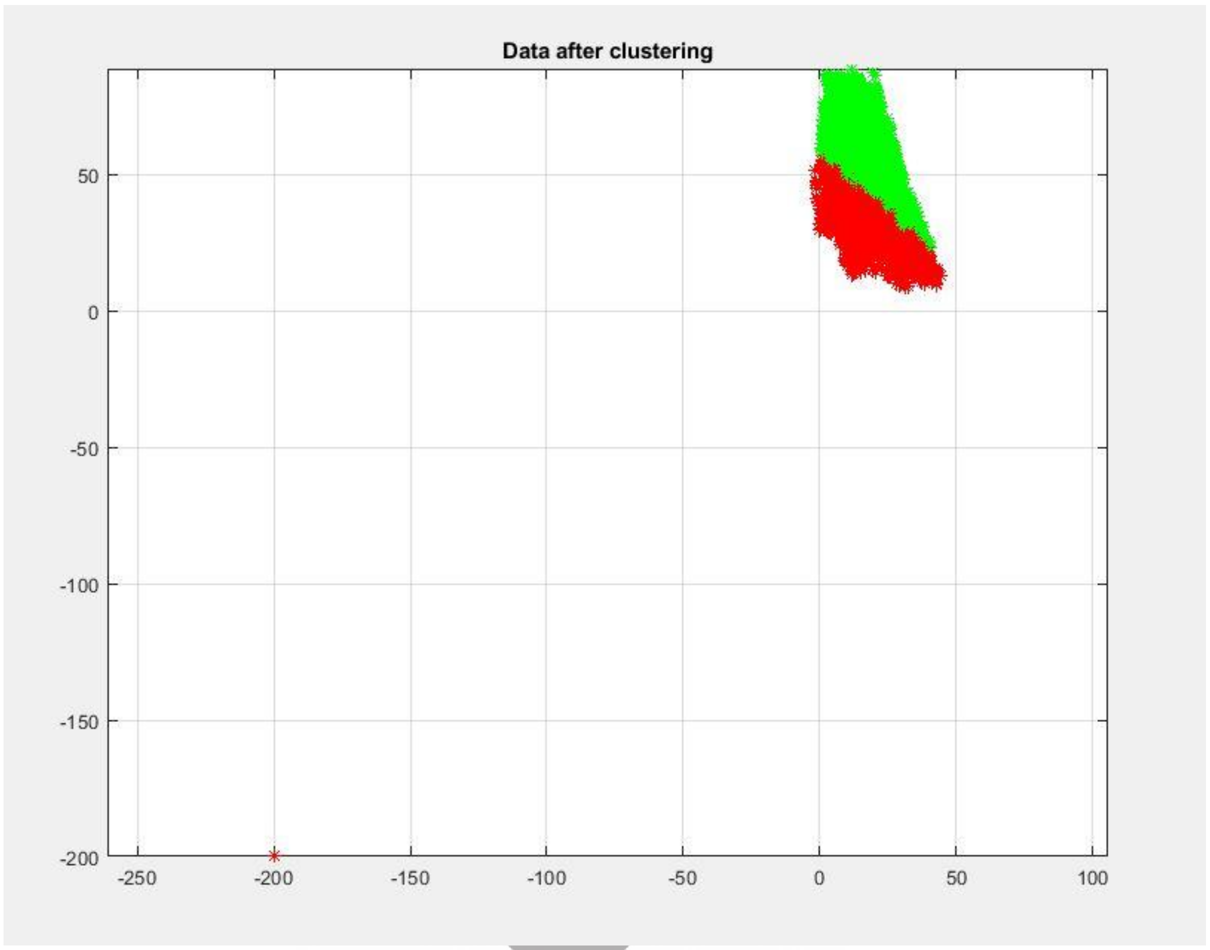


Fig 5.2 Data after Clustering

As shown in Fig 5.1 all the data belongs to one class or cluster, this is represented in the figure in green colour. After clustering input data using the K-Means algorithm, data was split into two groups, one shown in green and the other in red. This shows that this particular data belongs to class 1 and the other data belongs to class 2 respectively.

After running the Clustering file, data was saved to an external csv file with each row having a cluster id which determines which cluster each row belongs to. Saved data then became ready for classification. For the purpose of this study, 6177 samples were used for training and 3181 used for testing. After running the classifier on test data, the model reported the evaluation parameters of the confusion matrix, as shown in table 5.1.

Table 5.1 Confusion Matrix

Test Data = 3181	Predicted Anomalies in Data	Predicted Normal Data
Actual anomalies(1182)	TP=1123	FN=59
Actual Normalies (1999)	FP=52	TN=1947

From the Confusion matrix in Table 5.1 we note that the test-set consists of 3181 instances or records and from those 3181 instances, the actual normal data is 1999 instances consisting of 52 false positives (FP) and 1947 true negatives (TN) and the abnormal data has 1182 instances consisting of 1123 true positives (TN) and 59 false negatives (FN).

The model was tested using labelled dataset consisting of 3181 records, which consisted of 34% of the original input dataset. Each record in testing dataset constitutes three values; Temperature, Humidity and ID. The Temperature and Humidity are the readings that were originally obtained from the IoT sensors for Air Quality dataset (see chapter 4). The Clustering ID is a binary integer value which can only be either "1" or "2". These particular values represent the cluster to which each recording (i.e. Temperature and Humidity) belong to. The Clustering ID values were obtained as a result of running the K-Means algorithm 10 times against input dataset which consists of 9358 records.

C4.5 decision tree was trained using labelled training dataset consisting of 6177 records (i.e. 66% of input dataset). It then was tested against testing dataset, without giving it the clustering id which shows which cluster each of the records in the testing dataset belong to. In other words, C4.5 was tested against test dataset that does not contain labels. Since the number of records in the unlabelled test dataset is 3181, the model produced 3181 predictions in a form of "1" and "2". The resulting predictions were compared with the clustering id of each of the records in the labelled test dataset, and the results are show below:

The total number of **predicted anomalies** that were **actually anomalies** was 1123. Therefore, the true positive (TP) value of the model is:

$$TP = 1123$$

The total number of **predicted normalities** that were **actually normalities** was 1947. Therefore, the true negative (TN) value of the model is:

$$TN = 1947$$

The total number of **predicted anomalies** that were **actually normalities** was 52. Therefore, the false positive (FP) value of the model is:

$$FP = 52$$

The total number of the **predicted normalities** that were **actually anomalies** was 4. Therefore, the false negative (FN) value of the model is:

$$FN = 59$$

For one to be able to calculate the performance metrics for the proposed model such as overall accuracy, sensitivity, misclassification rate, prevalence, specificity and precision, the following actual values are important:

- The total number of records in the testing dataset = 3181
- The total number of normalities in the testing dataset = 1999
- The actual number of anomalies in the testing = 1182

Therefore, to calculate the overall accuracy of the proposed model we use:

$$\boldsymbol{Accuracy = \frac{TP + TN}{TP + FP + TN + FN}}$$

$$= \frac{1123+1947}{3181} = 0.9651$$

The overall accuracy of the proposed model is 0.9651.

The misclassification rate of the model, in producing the wrong prediction is calculated as follows:

$$\boldsymbol{Misclassification = \frac{FP + FN}{TP + FP + TN + FN}}$$

$$= \frac{52+59}{3181} = 0.03489$$

After calculation, the misclassification rate for the proposed model is equal to 0.03489.

The detection rate or precision of the proposed model is calculated as follows:

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{number\ of\ predicted\ abnormal\ records\ in}(\boldsymbol{TP}+\boldsymbol{FP}+\boldsymbol{TN}+\boldsymbol{FN})}$$

$$= \frac{1123}{1182} = 0.950$$

The False Positive rate of the proposed model is:

$$\boldsymbol{FP\ Rate} = \frac{\boldsymbol{FP}}{\boldsymbol{number\ of\ normal\ records\ in}\ (\boldsymbol{TP}+\boldsymbol{FP}+\boldsymbol{TN}+\boldsymbol{FN})}$$

$$= \frac{59}{1999} = 0.00291$$

The prevalence of the model:

$$\frac{\boldsymbol{actual\ number\ of\ abnormal\ records\ in}\ (\boldsymbol{TP}+\boldsymbol{FP}+\boldsymbol{TN}+\boldsymbol{FN})}{\boldsymbol{TP}+\boldsymbol{FP}+\boldsymbol{TN}+\boldsymbol{FN}}$$

$$= \frac{1182}{3181} = 0.372$$

The specificity of the model in reading and identifying normal readings:

$$\boldsymbol{TN} = \frac{\boldsymbol{TN}}{\boldsymbol{number\ of\ actual\ normal\ records\ in}(\boldsymbol{TP}+\boldsymbol{FP}+\boldsymbol{TN}+\boldsymbol{FN})}$$

$$= \frac{1947}{1999} = 0.974$$

5.1.1 Choice of Evaluation Methodology

As discussed in Chapter 4, for the purpose of this study we use 4 Evaluation Methodologies. To evaluate the classification accuracy of the proposed model, K-Means Clustering and C 4.5 Decision Tree are used. Furthermore, Naïve Bayes and Multi-Perceptron Artificial Neural Network are used. The Evaluation Methodologies chosen are the Hold out 66/34, the Hold out 50/50, 10-Fold Cross Validation and 5-Fold Cross Validation. Table 5.2 provides results and analysis of the performance of our model compared to the other two algorithms based on the evaluation method chosen for this experiment, a common discretization method in the form of Equal Frequency with 10 bins is used.

Table 5.2 Classification Accuracy based on Evaluation Methodology

ML Algorithm	5-Fold CV	10-Fold CV	Holdout -50/50	Holdout 66/34
C 4.5 + K-Means	87.79	87.83	87.64	87.52
Naïve Bayes	86.53	86.48	85.83	86.07
MLP ANN	94.44	93.40	91.96	89.69

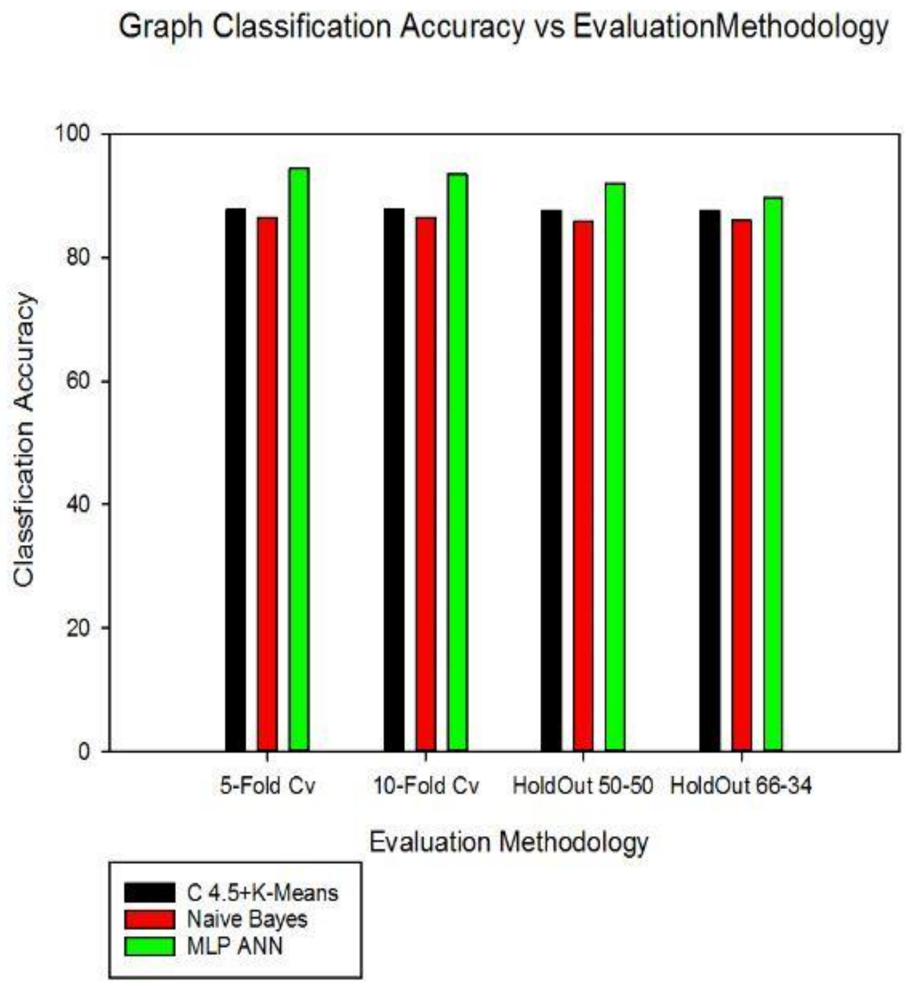


Fig 5.3 Classification Accuracy vs Evaluation Methodology

Table 5.2 and Fig 5.3 depict the classification accuracy of our proposed model compared with other classification algorithms based on choice of evaluation methodology. As can be noted from the diagram above, the Artificial Neural Network has a higher classification rate than the Naïve Bayes and the proposed model regardless of the evaluation methodology chosen. We ran a Friedman's ANOVA test to analyse if there is a variance of statistical significance due to evaluation methodology chosen. The results of the Friedman's test gave a value of $\boldsymbol{p = 0.06}$ i.e.$\boldsymbol{p > 0.05}$, which means we must retain the null-hypothesis where was the hypothesis stated, I might have missed it but I browsed through chapter 1 and could not find any stated hypothesis as the distribution from each evaluation is similar or the same. However from the Friedman's individual mean ranks we notice that there is no statistical difference between the 5-Fold Cross Valuation and the 10-Fold Cross Valuation. However, there is a difference between the 2 sets of Cross-Validation and the Hold out Evaluation Methodologies, this is shown in Appendix B. A Wilcoxon post-hoc test is done and it proves that although the distribution is the same between 2 Holdout methodologies, it is different between the Holdout and Fold Cross Validation methodologies.

5.1.2 Sample Size

Of the 9358 samples used in this dataset, 6200 which is 66% were dedicated for training. If the classification performance remains invariant while the 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 the intrusion detection model. The dataset is divided into ten subsets of 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100%. The purpose of dividing the size of the dataset set is crucial in inducting accurate probability estimates. Therefore we chose to start to generate the training set size at 10% of the total dataset so as to have an equal and normal distribution over 10 different points.

Table 5.3 Classification Accuracy on Various training sets.

Percentage	C 4.5 + K-Means	Naïve Bayes	MLP ANN
10%	95.00%	86.35%	95.93%
20%	96.30%	86.93%	96.24%
30%	96.33%	87.06%	96.36%
40%	96.31%	87.17%	96.31%
50%	96.40%	87.50%	96.36%
60%	96.44%	87.60%	96.33%
70%	96.40%	87.60%	96.3%
80%	96.70%	87.90%	96.7%
90%	97.75%	87.71%	97.11%
100%	98.9%	88.38%	100%

In table 5.3 the proposed model and the comparison algorithms are trained with varying training set sample sizes to determine the effect of the amount of training data on the overall classification accuracy of the algorithms.

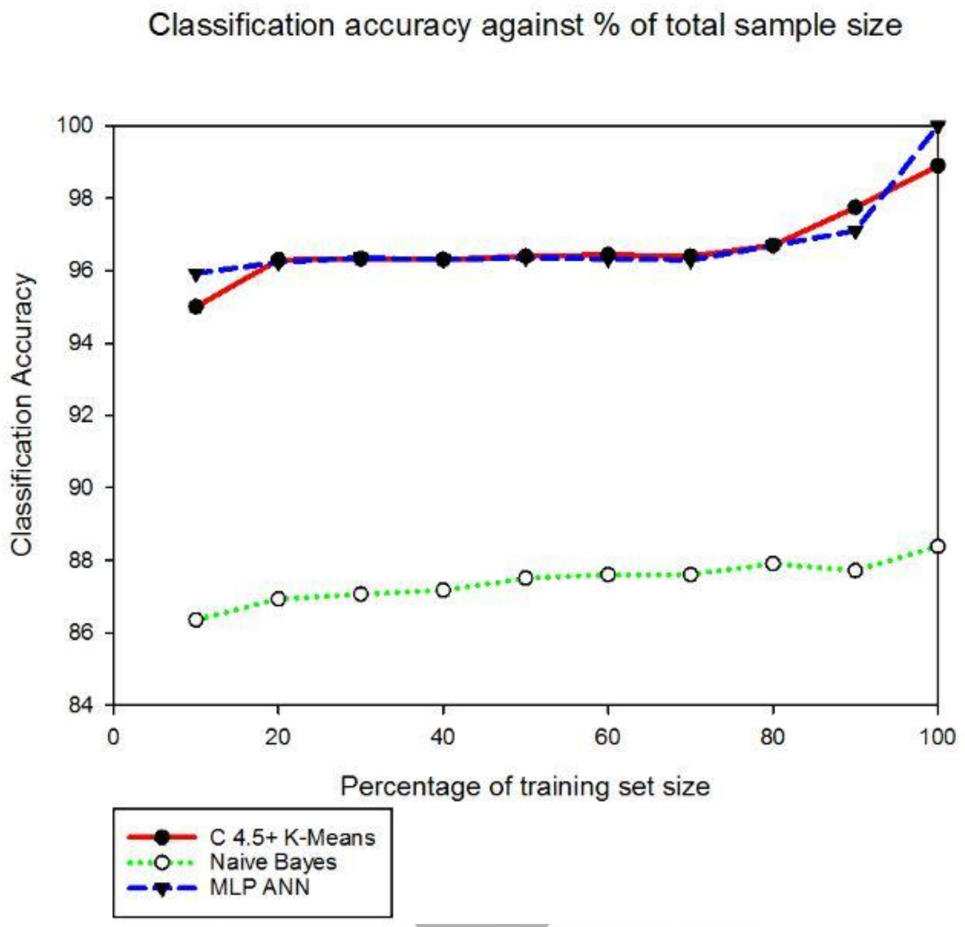


Fig 5.4 Classification Accuracy on Various training sample size

Fig 5.4 depicts the classification accuracy on various training sample size. Friedman's ANOVA test was used to test if there was a statistical significance in using different training sample size. The Friedman's test gave $p = 0.03$ which is $\boldsymbol{p} < 0.05$, this shows there is a significant difference with change of sample size. The details are shown in Appendix C.

5.1.3 Discretization Approach

Discretization is a process that occurs during data pre-processing as discussed in Chapter 3. There are a number of discretization approaches or algorithms. For the purpose of this study, we used 3 discretization algorithms to process our input data, these were the Equal Width (EW), Equal Frequency (EF) and the Proportional k-Interval Algorithm (PKI).

In our first experiment, the 3 classifiers being explored were trained with an equal fixed training sample size of 66%, the remainder of the data set was used for testing the classification accuracy of the algorithms.

Table 5.4 Classification Accuracy based on Discretization Approach.

Classifier	EW	EF	PKI
C 4.5 + K-Means	81.64%	96.51%	87.52%
Naïve Bayes	81.64%	87.43%	86.07%
MLP ANN	79.34%	96.51%	89.69%

In Table 5.4, classification accuracy of each classifier is represented based on 3 discretization approaches. According to the experiment, with equal width discretization (EW), the proposed model (C 4.5 and K-Means) and the Naïve Bayes Classifier have a higher classification accuracy (81.64%) as compared to the MLP ANN. However with equal frequency discretization (EF) and Proportional k-Interval discretization (PKI) the MLP ANN averages a higher classification accuracy as compared to the other 2 classifiers with 96.51% and 89.69% respectively.

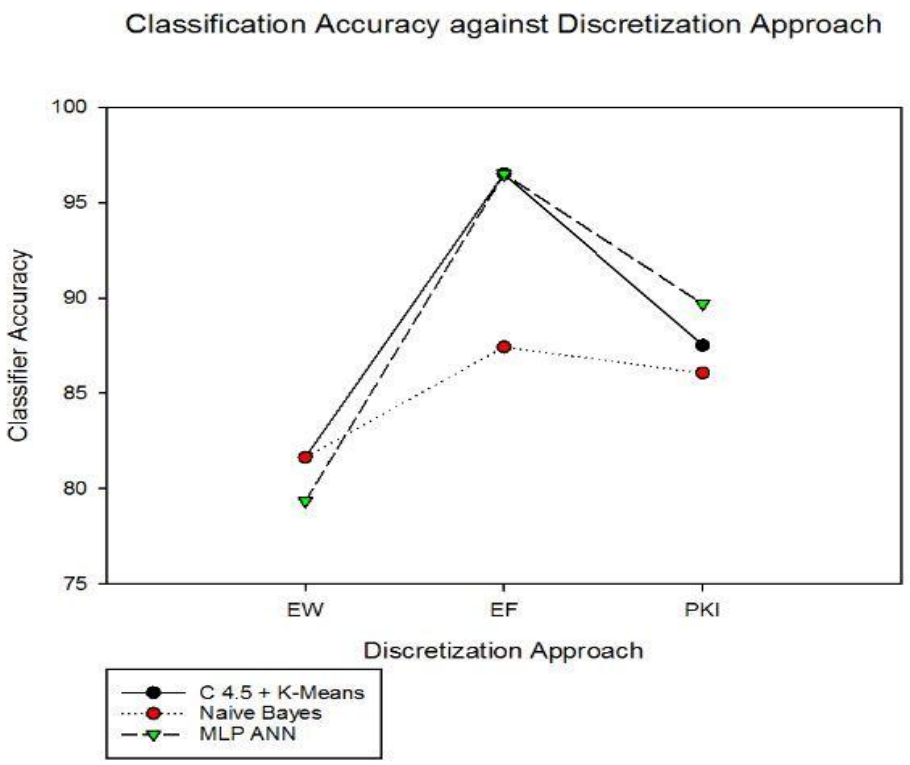


Fig 5.5 Classification Accuracy based on Discretization Approach.

Fig 5.5 depicts classification accuracy as per discretization approach. As shown, the MLP Artificial Neural Network and the proposed model that uses C 4.5 and K-Means are on average more accurate in classification more that the Naïve Bayes, with the Naïve Bayes classifier only more accurate than the MLP ANN when using equal width discretization (EW).

According to the Friedman test as detailed in Appendix D, there was a statistically significant difference in classification performance of the classifiers over different discretization approaches, using the value of p= 0.050. To confirm the Friedman's test outcome, the study conducted multiple comparisons using other methods. Wilcoxon Post host test was used for pairwise comparisons as detailed in Appendix D. There are 3 pairs of comparison, (Equal Frequency – Equal Width; Equal Frequency – K-Proportional Interval; K-Proportional Interval – Equal Width) all 3 pairs show a statistical significant difference with change of discretization approach as $\boldsymbol{p > 0.05}$ **(**$\boldsymbol{Z = -1.604; p = 0.109}$**)**.

Table 5.5 C 4.5 + K-Means Classification Accuracy on various discretization approaches

Percentage	EW	EF	PKI
10%	82.48	94.95	87.23
20%	82.27	92.29	87.64
30%	82.28	96.34	87.56
40%	82.17	96.31	87.69
50%	82.09	96.37	87.62
60%	81.88	96.45	87.41
70%	81.83	96.40	87.49
80%	82.36	96.69	88.24
90%	81.62	97.76	86.75
100%	78.79	98.94	90.42

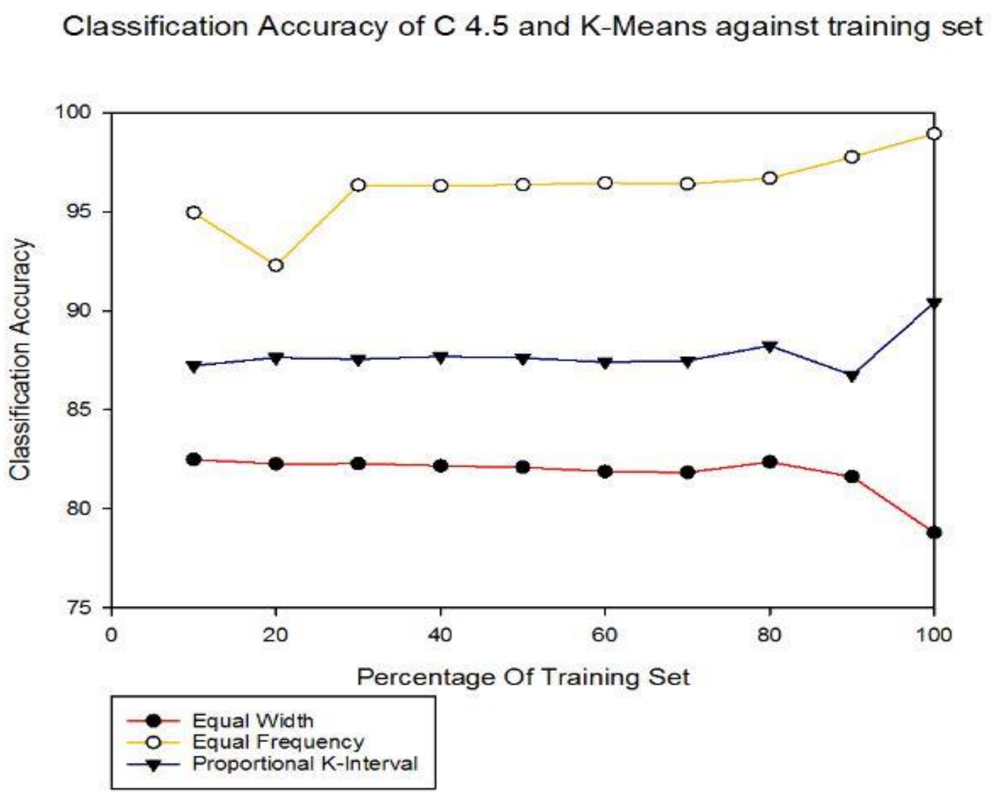


Fig 5.6 Classification Accuracy of C.45 + K-Means on different sample size.

Table 5.6 Classification Accuracy of Naïve Bayes on different sample size.

Percentage	EW	EF	PKI
10%	82.42	86.94	83.15
20%	82.27	88.18	84.58
30%	82.29	87.07	85.34
40%	82.17	88.90	85.68
50%	82.09	87.52	85.85
60%	81.88	87.12	85.81
70%	81.83	87.67	86.46
80%	82.36	87.97	86.64
90%	81.62	87.71	86.43
100%	78.79	88.29	87.23

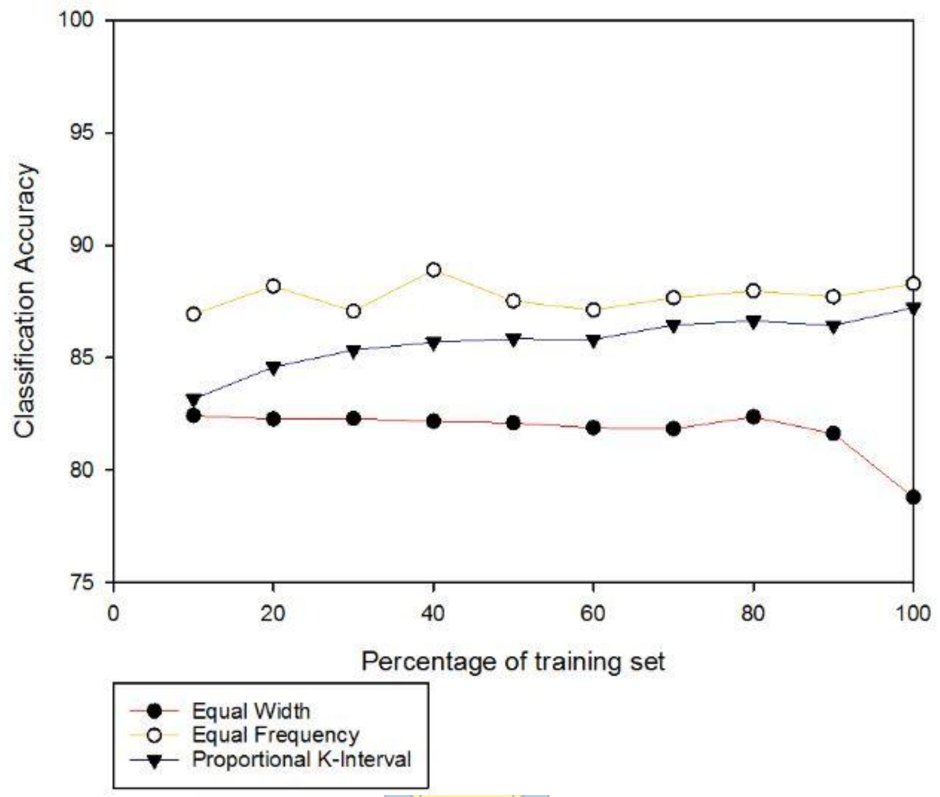


Fig 5.7 Classification accuracy of Naïve Bayes on different sample size

Table 5.7 Classification accuracy of MLP ANN on different sample sizes

Percentage	EW	EF	PKI
10%	79.94	95.94	89.53
20%	82.27	96.29	91.08
30%	82.29	96.37	92.21
40%	79.78	96.31	92.34
50%	79.71	96.37	91.96
60%	81.88	96.34	89.69
70%	81.83	96.26	90.07
80%	82.36	96.69	91.45
90%	81.62	97.44	91.23
100%	79.78	100	90.33

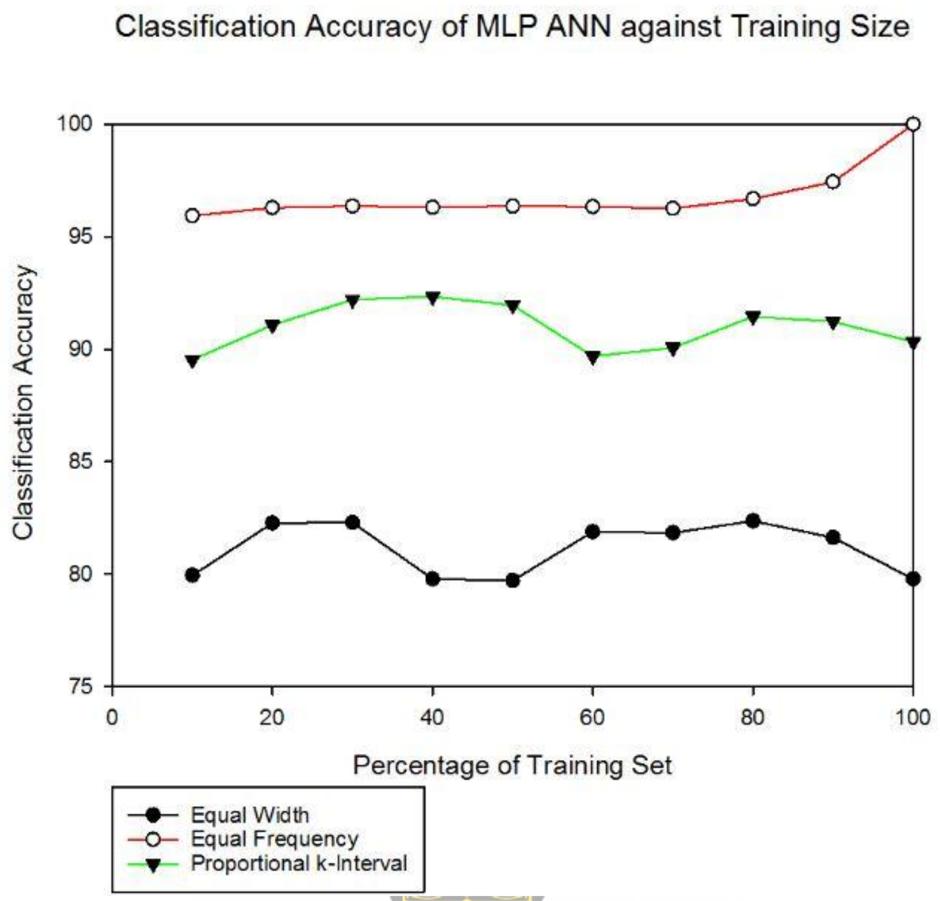


Fig 5.8 Classification Accuracy MLP ANN on different sample size

5.1.4 Speed of Classification

During the process of training, a classifier is trained on a certain sample size of data. During this process a classification algorithm builds a model based on binary rules, this model then gets used during the testing. . Most IoT devices have little computational power, and they are deployed in critical system such as heart rate sensors and gas monitors hence efficiency and high availability are crucial. When designing, developing and deploying an anomaly detection model for IoT, it is imperative that the model be fast, efficient and use as little computation power as possible. The study conducted an experiment on the efficiency and speed of learning for 3 classifiers listed in Table 5.8. This experiment uses a fixed discretization approach in the form of the Equal Width discretization algorithm. . The time taken to build the model for each of the classifiers at different evaluation methodologies was recorded, the results are shown in Table 5.8.

Table 5.8 Speed of Model Building and Testing

Evaluation Methodology	C 4.5 + K-Means	Naïve Bayes	MLP ANN
Holdout 66/34	0.01	0.01	33.65
Holdout 50-50	0.05	0.03	35.62
5- Fold Cross Validation	0.05	0.03	35.33
10-Fold Cross Validation	0.02	0.01	34.76

As it can be noticed from table 5.8, the Multi-Layer Perceptron Neural Network takes a considerable longer amount of time to build models and classify data as compared to the Naïve Bayes and C 4.5 and K-Means Algorithm, this is due to the complexity required in building an Artificial Neural Network, as shown in Appendix E. The experiment also gave solid evidence of the Speed of Classifiers as stated in Chapter 3 (see Table 3.6).

5.1.5 Receiver Operating Characteristics (ROC)

In the study we constructed an ROC Curve for the proposed model and then compared it with ROC Curve of the Naïve Bayes Classifier and Multi-Perceptron Artificial Neural Network.

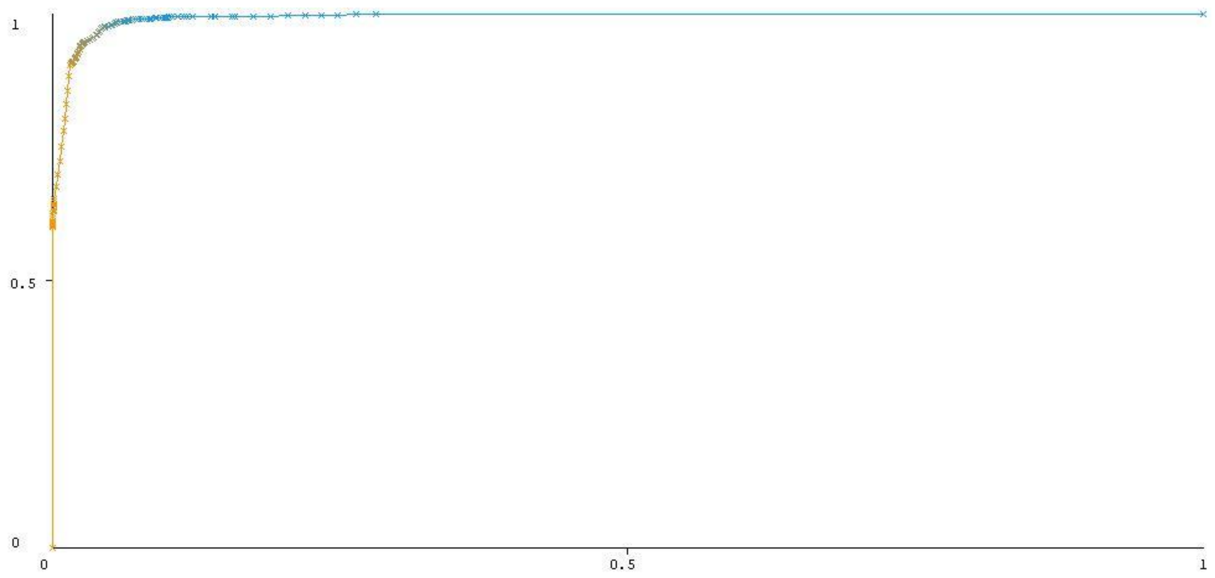


Fig 5.9 ROC C45+K- Means Algorithm

To construct the ROC Curve we trained the dataset with a 10-Fold Cross Validation Method, the Area under the Curve for our particular model is 0.9929 which shows that our model is highly accurate as the maximum threshold can only be 1.

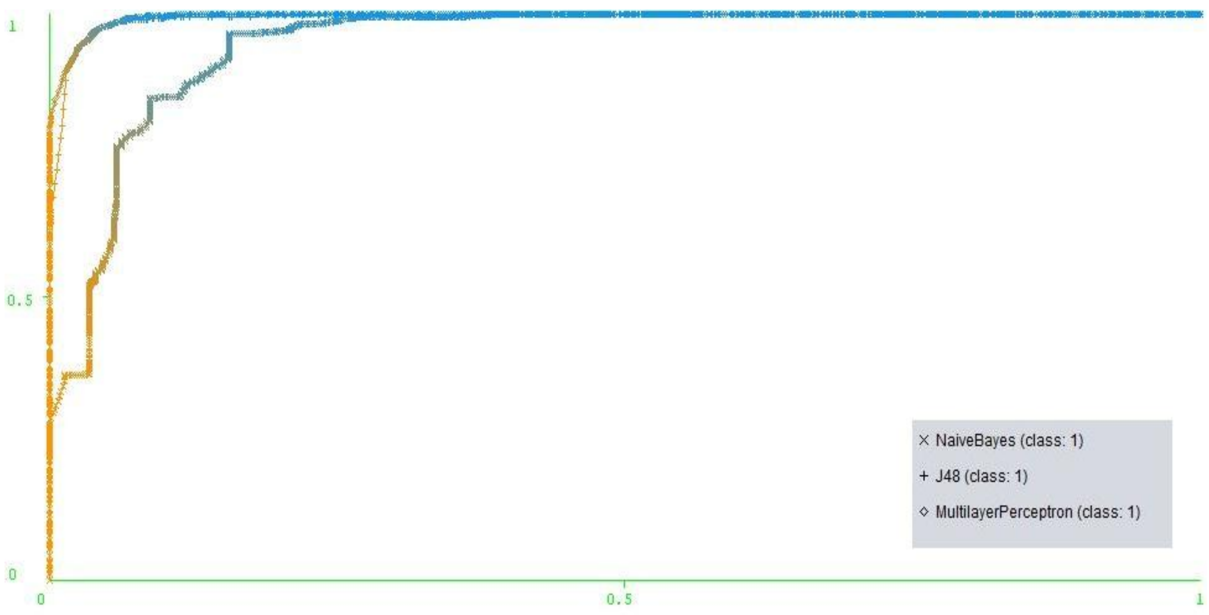


Fig 5.10 Comparison of ROC Curves of Multiple Classifiers.

In Fig 5.10 we compare the ROC Curves of the Naïve Bayes and Multi-Perceptron Artificial Neural Network with the proposed model. As can be noted, the Artificial Neural Network has a curve similar to the proposed model, and calculations reveal that its AUC is 0.9945 which is slightly better than that of our proposed model, which achieved **0.9929 as indicated earlier.** However, the Naïve Bayes Classifier falls short as Figure 5.10 shows.

5.2 Discussion and Analysis

The following subsection is dedicated to discussing the results presented in section 5.1. It should be noted that the results outlined are based on the effects of a particular parameter (Sample Size, Discretization, Speed and Type of Evaluation Methodology s) under investigation and also how it behaves varied while other parameters remain invariant. For instance, in one of the experiments conducted on how the choice of discretization approach, various methods were used to assess how they impact the classification performance while the factors such as evaluation methodology and training sample size 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 performance in classification for the proposed model.

5.2.1 Choice of Evaluation Methodology

In Machine Learning accuracy is the most widely used measure for performance evaluation of classification models. In this study we used different techniques to induce the model with training data from the dataset to build a model and also use test data on that particular model to test its classification performance. We made use of four evaluation methodologies, these were the Hold out 66/34, Hold out 50/50, the 10-Fold Cross Validation and 5 Fold Cross Validation. In the Hold out 66/34, the dataset during induction was divided into two parts, based on a 66% and 34% ratio, the 66% being used for training and building the model, while the rest was used for testing the model, in the Hold out 50/50, training data and testing data was divided equal parts from the original dataset. In k-Fold Cross Validation, data was divided into equal folds either 10 or 5 depending on the number of folds set during induction. An empirical experiment was conducted where we used a statistical approach in the Friedman's Analysis of Variance test to test if the different evaluation methodologies, had a statistical significance in the in the accuracy of the classifiers, the results of that are presented in an IBM SPSS file in Appendix B.

5.2.2 Size of Training Set/ Sample Size

This section dealt with the analysis of how varying training sample sizes affects the classification performance. In Classification against training sample size graph it can be observed that the classification accuracy chances with a change in training sample size, it is more vivid with the Naïve Bayes Classifier, as it gradually increases proportional to an increase in training sample size. From the observations there is generally an improvement in predictive classification performance as we traverse the graph from 10% to 100% of the total training set. By using Friedman's test we get a *p* value of less than 0.05 which shows that there is a statistic difference with change in training set or sample size.

5.2.3 Type of Discretization

This part of the empirical study analysed the impact of the choice of discretization algorithm on the classification performance. Figure 5.4 depicts the classification accuracy of the each classifier under study against the discretization method used in pre-processing prior to training. The different line graphs correspond to how the 3 classifiers behaved for the 3 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. It is noted that there is a statistical significance in accuracy carried with each discretization approach.

5.3.4 ROC Curves

Anomaly based intrusion detection models are characterized by their level of accuracy , hence most evaluation methods used in machine learning are based on classification accuracy, this approach yields accurate evaluation results, however there are problems with this particular approach;-

- There may be different costs associated with false positive and false negative errors.
- Training Data may not reflect true class distribution.

To solve these problems Machine Learning uses the Receiver Operating Characteristic (ROC) approach. ROC Curve is a graph that shows the performance of a classification model at all classification thresholds. ROC curves assess predictive behaviour independent of error costs and class distributions. The ROC Curve is constructed from True Positive Rate and False Positive Rate. The Area under the Curve (AUC-ROC) is a performance measurement for classification at various threshold settings. ROC is a probabilistic curve and the degrees of specificity is called AUC. The higher the AUC the better the model is at binary classification. In this study a ROC Curve was constructed for the proposed model and AUC was calculated. This particular ROC Curve is compared with the Naïve Bayes and MLP ANN, as shown in Fig 5.10, has a better specificity rate than the Naïve Bayes, but is similar with the Multi Perceptron Artificial Neural Network.

5.3 Conclusion

This chapter outlined the performance and accuracy of the proposed anomaly detection model under individual and collective configurations. Each experiment was described and discussed in detail and thereafter detailed analysis of the outcomes of the experiments was conducted. 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 Recommendations

Our research has been an investigation to answer the critical question: How can Machine Learning Algorithms be used as a solution to secure the Internet of Things? The study proposes an anomaly detection model, which can be used as a form of Intrusion Detection Mechanism. However the question remains: How resilient and effective is this solution as compared to other solutions, to answer this question an empirical investigation was done to investigate how the size of a training set, type of discretization approach and learning method individual and collectively affect the classification performance of this model in distinguishing between normal and anomaly data? We made use of a generic dataset measuring Air Quality from the UCI repository 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 Purpose and Findings

The purpose of this study is to analyse how various threats can impact security and privacy in the Internet of Things, and to develop an effective lightweight Intrusion and Anomaly Detection Model for IoT to help detect threats in the environment. The study seeks to employ Machine Learning and Data Mining algorithms to implement an anomaly detection model for the Internet of Things. The ultimate quest of the machine learning and data mining community is to implement optimal performing classification models. Therefore we proposes use of the C 4.5 Decision Tree and K-Means Clustering Algorithms as an anomaly detection model in data being transmitted in between devices in an IoT network. In the study we hypothesized the impact of speed of learning, training sample, training methodology and sample size.

The proposed model which uses C 4.5 Decision Tree and K-Means Clustering has an accuracy of 96.51%, falls short of related work done by E. Hodo et al (2016) which has a classification accuracy of 99.4%. Hodo suggests the use of a Multi-Perceptron Layer Artificial Neural Network in detecting DoS on Network Layer Systems. However Hodo et al (2016) does not take into account the computational costs of training and testing a MLP ANN classifier. Similar work was conducted by Granjal, Silva and Lourenço (2018) who propose an anomaly detection system for the application layer using Support Vector Machines (S.V.Ms), their experiment had an accuracy of 93% which falls short of the proposed model in-terms of classification accuracy. The domain for the application of this experiment is in intrusion detection for the Internet of Things.

6.2 Empirical findings vs Research questions

1) What are the security challenges and vulnerabilities being faced in IoT security implementations and what tentative security measures can be used help improve security?

The research introduced the concept of IoT in Chapter 2. The Internet of Things is used in areas such as health care, home automation, smart cities, modern agriculture which are making our everyday life easier. However, there exist challenges in terms of security that come about with this technology, as there is no set standard architecture for this technology with manufacturers keen on producing their own various objects for financial gain, as security is not the main concern. Attacks such as DoS and DDoS have arisen an example is the Mirai DDoS Malware that used IoT security vulnerabilities to render a number of Fortune 500 websites being unavailable such as Netflix and Twitter.

2) Can Data mining and Machine learning techniques be used as an effective way to develop a comprehensive defence mechanism against attacks on an IoT Network?

The research firstly discussed the concepts of Machine Learning Algorithms in Chapter 3. A few definitions were presents to explain the concept. The use of Machine Learning algorithms 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.

3) Can Data mining and Machine learning techniques be used as an effective way to develop a comprehensive defence mechanism against attacks on an IoT Network?

In Chapter 3, under Related Work, we review research work done by previous academia in using Machine Learning approaches in Intrusion detection and at the beginning of Chapter 3, when introducing the concept of Machine Learning, we review a number Machine Learning algorithms and how they are trained and tested in order to classify data, which is the core functionality of an intrusion detection system i.e. to classify data as normal and abnormal.

4) Which Machine learning and Data mining algorithms can be best implemented to effectively develop an efficient self-learning program/system to detect anomalies in data being transmitted in an IoT ecosystem?

In Chapter 3, we dived into Machine Learning, and reviewed a number of Machine Learning Algorithms. We compared the algorithms strengths and weaknesses and how they can be can they be implemented as in an anomaly detection model specifically for IoT. During Implementation we compared the Multi-Layer Perceptron Artificial Neural Network (MLP ANN), Naïve Bayes and the proposed model which uses K-Means Clustering and C 4.5 Decision Tree Algorithm. The algorithms' performance metrics where measured, and evaluation methodologies where implemented to determine which of these algorithms can be used to train and test a model on a given dataset more effectively. From the findings of the research the K-Means and C 4.5 Decision Tree Algorithms proved to be accurate and more efficient in terms of false positive rate and speed of training and testing than its competition on this particular instance.

6. 3 Implications and Limitations

This study supports the notion that machine learning and data mining algorithms are the most effective method of designing and implementing anomaly detection models for the Internet of Things. However the study is limited to designing a model that can be used for anomaly detection using machine learning and data mining algorithms and does not focus on improving the algorithms themselves. The model is designed to be implemented for data exchange between IoT devices at application level. Lightweight algorithms in the form of K-Means and C 4.5 Algorithm can be trained at relatively low computation cost as compared to more complex algorithms such as the MLP ANN as shown in Chapter 5 while testing for speed of learning. However the MLP ANN is more accurate at detecting anomalies, hence in areas were computational costs are not a cause for concern, it might be deployed as is the case with Hodo el at' (2016) experiment (Section 3.8)

6.4 Recommendations and Future work

The proposed model makes use of data mining and machine learning algorithms to detect anomalies in IoT data at application level. This model can be implemented in IoT systems that can tolerate a misclassification rate of 2.1% where anomalies that remain undetected do not to lead to catastrophic scenarios such as loss of life, data and assets. Throughout the course of this study it is noted that generally Artificial Neural Networks have a higher classification rate of any of the compared classifiers in Chapter 5 its only achilles heel being the speed of training and testing, as it is a complex algorithm making it computationally expensive.

6.5 Conclusion

Machine Learning Algorithms have been used in a wide-range of applications. The focal point of this research has been to apply them and Data Mining techniques in detecting DoS attacks in IoT environments. Literature provides numerous advantages that make Machine Learning Algorithms the best techniques to be used in anomaly detection, they have proven to be excellent in classification, clustering and regression tasks. This research embarked on the analysis of the impact of evaluation models, training sample size and discretization algorithms on the classification performance of classifiers in detecting potential attacks in IoT environments. The hope of the researcher is that this study will provide sound ecosystem modelling choices for researchers intending to apply machine learning in any field of study. Poor choices of the sample size, discretization technique will have adverse effects on the classification accuracy of chosen classifier.

References

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

Aggarwal, Charu C., Naveen, A., & Amit, S. 2013. *The Internet of Things: A Survey from the Data-Centric Perspective*, *Managing and Mining Sensor Data*.

Alkasassbeh, M., Al-Naymat ,G., Hassanat, A. & Almseidin, M. 2016. Detecting Distributed Denial of Service Attacks Using Data Mining Techniques. *(IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 1, 2016.*

An Internet of Things Reference Architecture. 2016. 1st ed. Symantec, pp.1-22.

Attaway, S. 2013. Matlab: A Practical Introduction to Programming and Problem Solving (3rd Eds.). Butterworth-Heinemann.

Atzori, L., Iera, A. & Morabito, G. 2010. The internet of things: a survey, Comput. Netw. 54(15) (2010) 2787–2805.

Badr . A comparative Study of decision tree ID3 and C 4.5. *(IJACSA) International Journal of Advanced Computer Science and Applications, Special Issue on Advances in Vehicular Ad Hoc Networking and Applications.*

Bell, J. 2014. Machine Learning: Hands-On for Developers and Technical Professionals. John Wiley & Sons.

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).

Buczak, A. L. & Guven, E. 2015. A Survey of Data Mining and Machine Learning

Burbidge, R. & Buxton, B. An Introduction to Support Vector Machines for Data Mining. UCL: Computer Science Dept. 2001.

Butun, I., Morgera, S. & Sankar, R. 2013. A Survey of Intrusion Detection Systems in Wireless Sensor Networks. *Communications Surveys Tutorials, IEEE.*

Chen, F., Deng, P., Wan, J., Zhang, D., Vasilakos, A. & Rong, X. 2015. Data Mining for the Internet of Things: Literature Review and Challenges. International Journal of Distributed Sensor Networks. 2015;2015:1-14.

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

Farooq, M., Waseem, M., Khairi, A. & Mazhar, S. 2015. A Critical Analysis on the Security Concerns of Internet of Things (IoT). *International Journal of Computer Applications*, 111(7), pp.1-6.

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.

Gartner .2013. Forecast: The Internet of Things, Worldwide, 2013. Retrieved from: http://www.gartner.com/document/2625419?ref=QuickSearch&sthkw=G00259115.

Hajare, S. A. 2016. Detection of Network Attacks Using Big Data Analysis. In International Journal on Recent and Innovation Trends in Computing and Communication, vol. 4, issue 5, pp. 86-88.

Haunsheng, N. & Hong L. 2012. Advances in Internet Of Things Vol.2.

Hodo, E. 2016. Threat analysis of IoT networks using artificial neural network intrusion detection system. In International Symposium on Networks, Computers and Communications (ISNCC), pp. 1-6.

Jian, A., Xiao-Lin, G. & Xin H. ”Study on the Architecture and Key Technologies for Internet of Things,” in Advances in Biomedical Engineering, Vol.11, IERI-2012, pp. 329-33

Jog, V. & Murugan, T. 2016. A Critical Analysis on the Security Architectures of Internet of Things: The Road Ahead. *Journal of Intelligent Systems*.

Jung-Tae, K. 2016. Analyses and Security Requirements for Smart Home Network Based on Internet of Things.

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

Kozlov, D., Veijalainen, J. & Ali, Y. 2012. *Security and Privacy Threats in IoT Architectures*.

Lee, C.H. 2007. A Hellinger-based discretization method for numeric attributes in classification learning. *Knowledge-Based Systems.*

Lin, H. & Bergmann, N. 2016. IoT Privacy and Security Challenges for Smart Home Environments. *Information*, 7(3), p.44.

Mahdavinejad. 2015. Machine Learning for the Internet of Things: a survey.

Methods for Cyber Security Intrusion Detection. In IEEE Communications Surveys & Tutorials, vol. 18, issue 2, pp. 1153-1176.

Pajouh, H. H., Javidan, R., Khayami, R. & Ali, D. 2016. A Two-layer Dimension Reduction and Two-tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks.

Patel, K.2016. Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges International Journal of Engineering Science and Computing, May 2016.

Philokypros, P. 2018. A Signature-based Intrusion Detection System for the Internet of Things.

Roman, R., Zhou, J. & Lopez, J. 2006. Applying intrusion detection systems to wireless sensor networks. In *Proceedings of IEEE Consumer Communications and Networking Conference*, pages 640–644, 2006.

Sai Ram, K. & Gupta, A. 2016. IOT based Data Logger System for weather monitoring using Wireless sensor networks. *International Journal of Engineering Trends and Technology*, 32(2), pp.71-75.

Sang, Y. & Gao, X. 2013. Security Issues and Protective Measures of the Internet of Things Architecture. *Advanced Materials Research*, 765-767, pp.1007-1010.

Security Intelligence. 2017. *Lessons from the Dyn DDoS Attack*. [Online] Available at: https://securityintelligence.com/lessons-from-the-dyn-ddos-attack/ [Accessed 1 Jul. 2017].

Shalev-Shwartz, S. & Ben-David, S. 2014. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press.

Sherasiya, T. & Upadhyay, H. 2016.Intrusion Detection System for Internet of Things. In IJARIIE-ISSN(O), vol. 2, issue. 3 (2016)

Shyu, M., Chen, S., Sarinnapakorn, K., & Chang, L. (2003). A novel anomaly detection scheme based on principal component classifier. In Paper presented at the proceedings of ICDM'03.

Tayyaba, K. 2017. IoT Security against DDoS attacks using Machine Learning Algorithms, International Journal of Scientific and Research Publications, Volume 7, Issue 6, June 2017.

Tsai.2009. Expert Systems with Applications 36 11994–12000

Vasilomanolakis, E., Daubert, J. & Luthra, M. (n.d.). *On the Security and Privacy of Internet of Things Architectures and Systems*.

Vapnik, V. 1998. Statistical learning theory. New York: John Wiley.

Vijayalakshmi, A. & Arockiam. L. 2016. – A Study on Security Issues and Challenges in IOT.

Wang, W., Guan, X., & Zhang, X. 2004. A novel intrusion detection method based on principle component analysis in computer security. In Paper presented at the proceedings of the international symposium on neural networks.

Weber .R, "Internet of Things–New security and privacy challenges," Elsevier Computer Law & Security Review, vol. 26, no. 1, pp. 23–30,2010.

Weiser, M. 1999. The computer for the 21st century, Mob. Computer. Communication. Rev. 3 (3) (1999) 3–11.

Zhao, K. & Ge, L. 2016. A Survey on the Internet of Things Security. In 9th International Conference on Computational Intelligence and Security (CIS). DOI: 10.1109/CIS.2013.145.

Zheng .Z, Wang .J, Zhu .Z, "A General Anomaly Detection Framework for Internet of Things," in Proc. 41st IEEE/IFIP International Conference on Dependable Systems and Networks, Hong Kong, June 27-30, 2011.

Appendix A – Matlab code for proposed model

Classification.m

```
Data = 'clusteredData.csv';
CData = csvread(Data);
%Ouput evaluation variables.

TP=0;
TN=0;
FP=0;
FN=0;
%take out data for training.

train_data = CData(1:6650,1:2)'
train_targets = CData(1:6650,3)'

%take out data for testing.
test_data = CData(6651:9357,1:2)'
test_targets = CData(6651:9357,3)'

%call C4.5 classification method to create the tree.
[tree, discrete_dim] = C4_5(train_data, train_targets, 1)
disp('Classify test samples using the tree')

%use the tree built using training data to examin the test data. Input: test data, number of
records in test
%data, tree, discrete_dim indicates number of classes i.e. 2, 1 or 2.

result = use_tree(test_data, 1:size(test_data,2), tree, discrete_dim, unique(train_targets));

%transpose the matrix result'
%restore the result to its orginial state.

result1 = result';
test_data=test_data'
test_targets1=test_targets'

[S,dim] = size(test_targets1);
desResult = zeros(S,1);
figure(1)
for i = 1:S
if test_targets1(i,1) == 1 && result1(i,1) == 1
TP = TP+1;
plot(test_data(i,1), test_data(i,2), '*g')
hold on
elseif test_targets1(i,1) == 2 && result1(i,1) == 2
TN = TN+1;
plot(test_data(i,1), test_data(i,2), 'om')
```

```
hold on
elseif test_targets1(i,1) == 1 && result1(i,1) == 2
FN = FN+1;
plot(test_data(i,1), test_data(i,2), 'xr')
hold on
elseif test_targets1(i,1) == 2 && result1(i,1) == 1
FP = FP+1;
plot(test_data(i,1), test_data(i,2), '+y')
hold on
end
end
title('Test data after classification')
grid on
axis equal
for i = 1:S
if result1(i,1) == 1
desResult(i,1) = 'A';
else
desResult(i,1) = 'N';
end
end
classifiedReslut = [test_data test_targets' result' ]
dlmwrite('classifiedReslut.txt', classifiedReslut, 'delimiter', '\t');
dlmwrite('NormalAbnormalclassifiedReslutReslut.txt', desResult, 'delimiter', '\t');
disp('True Negative')
TN
disp('True Positive')
TP
disp('False Negative')
FN
disp('False Positive')
FP
disp('Detection Rate')
DetectionRate = (TP) / (TP+FP)
disp('False Alarm')
FalseAlarm = (FP) / (FP+TN)
disp('Accuracy')
Accuracy = (TP+TN) / (TP+TN+FP+FN)
```

K.m

```
% This code takes as input parameters, the real dataset that we need to
% cluster and the number of clusters that we need. It returns the prototypes
% values

function [m] = KM(x,K)
MM=1;
NN=MM+2;
U=1;
UU=0;
```

```
[dataLen,dim]=size(x);
dataLen;
dim;
m = rand(K,dim)*1;
d=zeros(dataLen,K);
flag=zeros(dataLen,1);
targets = zeros(K,1);
%this is to run the K-means 10 times.

for k=1:K
for n=1:dataLen
d(n,k)=norm(x(n,:)-m(k,:));
end
end
[mins,I] =min(d');
I(1,1)
d;
for n=1:dataLen
flag(n) = 1;
end
%trying to find best represntative clusters.
flag;
for k=1:K
num=0;
den=0;
for n=1:dataLen
val1=0;
val2=0;
if ((d(n,k) - mins(1,n) < 0.00001) && flag(n,1) ==1)
% if(0)
flag(n,1)=0;
if mins(1,n) ~= 0
for L=1:K
val1 = val1 + d(n,L)^(-MM);
val2 = val2 + d(n,L)^(-MM);
end
val1=val1-mins(1,n)^(-MM);
val2=val2-mins(1,n)^(-MM);
val1 = val1 * NN * mins(1,n)^(NN-2);
val2 = val2 * NN * mins(1,n)^(NN-2);
val1= -1*(NN-MM) * mins(1,n)^(NN-MM-2) - val1;
val2= -1*(NN-MM) * mins(1,n)^(NN-MM-2) - val2;
else
val1 = mins(1,n)^(NN-MM-2)*(MM-NN);
val2 = mins(1,n)^(NN-MM-2)*(MM-NN);
end
val1 = val1 * x(n,:);
num = num + val1;
den = den + val2;
else
```

```
val1= val1 + x(n,:) * ( U* MM * mins(1,n)^(NN-0.01)/d(n,k)^(MM+2));
val2= val2 + ( U* MM * mins(1,n)^(NN-0.01)/d(n,k)^(MM+2));
num = num + val1;
den = den + val2;
end
end
m(k,:) = num/den;
targets(k,1)= I(1,1)
end
 targets
 m
```

Make_tree.m

```
function tree = make_tree(patterns, targets, inc_node, discrete_dim, maxNbin, base)
[Ni, L]           = size(patterns);
Uc          = unique(targets);
tree.dim          = 0;
tree.split_loc       = inf;
if isempty(patterns),
return
end
if ((inc_node > L) | (L == 1) | (length(Uc) == 1)),
H          = hist(targets, length(Uc));
[m, largest]   = max(H);
tree.Nf = [];
tree.split_loc = [];
tree.child     = Uc(largest);
return
end
for i = 1:length(Uc),
Pnode(i) = length(find(targets == Uc(i))) / L;
end
Inode = -sum(Pnode.*log(Pnode)/log(2));
delta_Ib = zeros(1, Ni);
split_loc  = ones(1, Ni)*inf;
for i = 1:Ni,
data  = patterns(i,:);
Ud = unique(data);
Nbins  = length(Ud);
if (discrete_dim(i)),
P  = zeros(length(Uc), Nbins);
for j = 1:length(Uc),
for k = 1:Nbins,
indices   = find((targets == Uc(j)) & (patterns(i,:) == Ud(k)));
P(j,k)   = length(indices);
end
end
Pk = sum(P);
P = P/L;
```

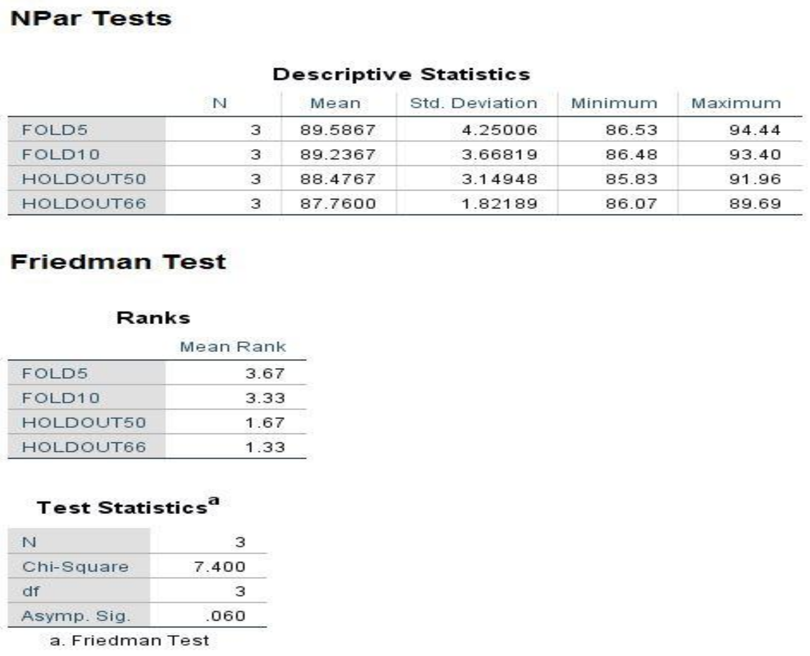
```
Pk = Pk/sum(Pk);
info = sum(-P.*log(eps+P)/log(2));
delta_Ib(i) = (Inode-sum(Pk.*info))/-sum(Pk.*log(eps+Pk)/log(2));
else
P  = zeros(length(Uc), 2);
[sorted_data, indices] = sort(data);
sorted_targets = targets(indices);
I  = zeros(1, L-1);
for j = 1:L-1,
   P(:, 1) = hist(sorted_targets(1:j) , Uc);
P(:, 2) = hist(sorted_targets(j+1:end) , Uc);
Ps    = sum(P)/L;
P    = P/L;
Pk = sum(P);
P1 = repmat(Pk, length(Uc), 1);
P1 = P1 + eps*(P1==0);
info  = sum(-P.*log(eps+P./P1)/log(2));
I(j)  = Inode - sum(info.*Ps);
end
[delta_Ib(i), s] = max(I);
split_loc(i) = sorted_data(s);
end
end
[m, dim] = max(delta_Ib);
dims = 1:Ni;
tree.dim = dim;
Nf    = unique(patterns(dim,:));
Nbins  = length(Nf);
tree.Nf = Nf;
tree.split_loc = split_loc(dim);
if (Nbins == 1)
H        = hist(targets, length(Uc));
[m, largest]   = max(H);
tree.Nf = [];
tree.split_loc = [];
tree.child     = Uc(largest);
return
end
if (discrete_dim(dim)),
for i = 1:Nbins,
indices = find(patterns(dim, :) == Nf(i));
tree.child(i)  = make_tree(patterns(dims, indices), targets(indices), inc_node,
discrete_dim(dims), maxNbin, base);
end
else
   indices1       = find(patterns(dim,:) <= split_loc(dim));
indices2       = find(patterns(dim,:) > split_loc(dim));
if ~(isempty(indices1) | isempty(indices2))
tree.child(1)  = make_tree(patterns(dims, indices1), targets(indices1), inc_node,
discrete_dim(dims), maxNbin, base+1);
```

```
tree.child(2)  = make_tree(patterns(dims, indices2), targets(indices2), inc_node,
discrete_dim(dims), maxNbin, base+1);
else
H       = hist(targets, length(Uc));
[m, largest]   = max(H);
tree.child     = Uc(largest);
tree.dim = 0;
end
end
```

Use_tree.m

```
function targets = use_tree(patterns, indices, tree, discrete_dim, Uc)
targets = zeros(1, size(patterns,2));
if (tree.dim == 0)
targets(indices) = tree.child;
return
end
dim = tree.dim;
dims= 1:size(patterns,1);
if (discrete_dim(dim) == 0),
in        = indices(find(patterns(dim, indices) <= tree.split_loc));
targets    = targets + use_tree(patterns(dims, :), in, tree.child(1), discrete_dim(dims), Uc);
in        = indices(find(patterns(dim, indices) > tree.split_loc));
targets    = targets + use_tree(patterns(dims, :), in, tree.child(2), discrete_dim(dims), Uc);
else
Uf        = unique(patterns(dim,:));
for i = 1:length(Uf),
if any(Uf(i) == tree.Nf)
    in= indices(find(patterns(dim, indices) == Uf(i)));
    targets  = targets + use_tree(patterns(dims, :), in, tree.child(find(Uf(i)==tree.Nf)),
discrete_dim(dims), Uc);
end
end
end
```

Appendix B



NPar Tests

Descriptive Statistics

	N	Mean	Std. Deviation	Minimum	Maximum
FOLD5	3	89.5867	4.25006	86.53	94.44
FOLD10	3	89.2367	3.66819	86.48	93.40
HOLDOUT50	3	88.4767	3.14948	85.83	91.96
HOLDOUT66	3	87.7600	1.82189	86.07	89.69

Friedman Test

Ranks

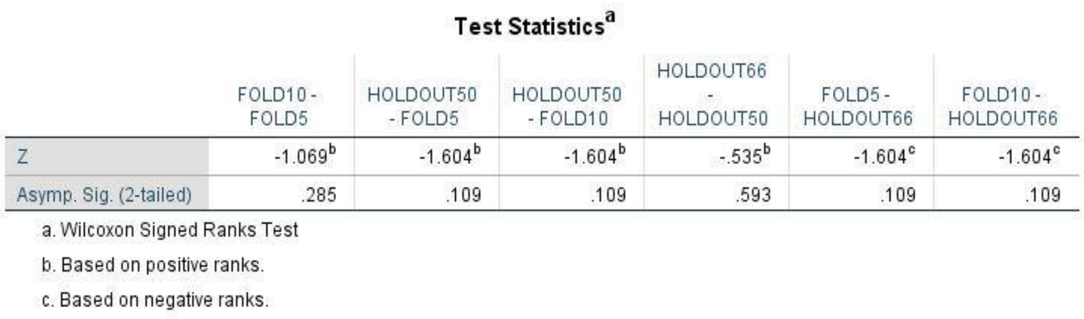
	Mean Rank
FOLD5	3.67
FOLD10	3.33
HOLDOUT50	1.67
HOLDOUT66	1.33

Test Statistics[a]

N	3
Chi-Square	7.400
df	3
Asymp. Sig.	.060

a. Friedman Test

Wilcoxon Test



Test Statistics[a]

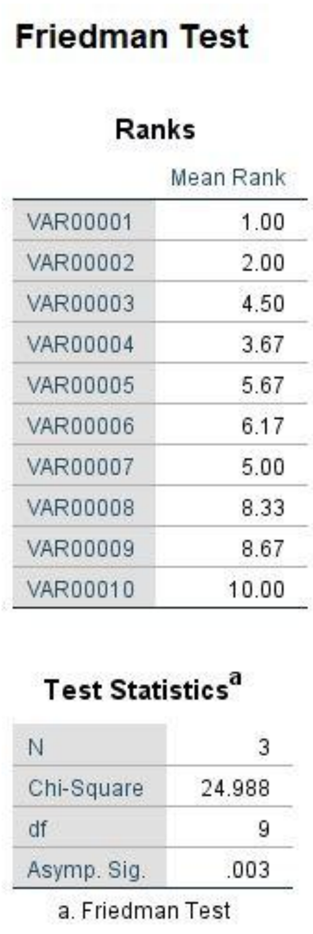
	FOLD10 - FOLD5	HOLDOUT50 - FOLD5	HOLDOUT50 - FOLD10	HOLDOUT66 - HOLDOUT50	FOLD5 - HOLDOUT66	FOLD10 - HOLDOUT66
Z	-1.069[b]	-1.604[b]	-1.604[b]	-.535[b]	-1.604[c]	-1.604[c]
Asymp. Sig. (2-tailed)	.285	.109	.109	.593	.109	.109

a. Wilcoxon Signed Ranks Test

b. Based on positive ranks.

c. Based on negative ranks.

Appendix C – Friedman ANOVA Test on Sample Size



Friedman Test

Ranks

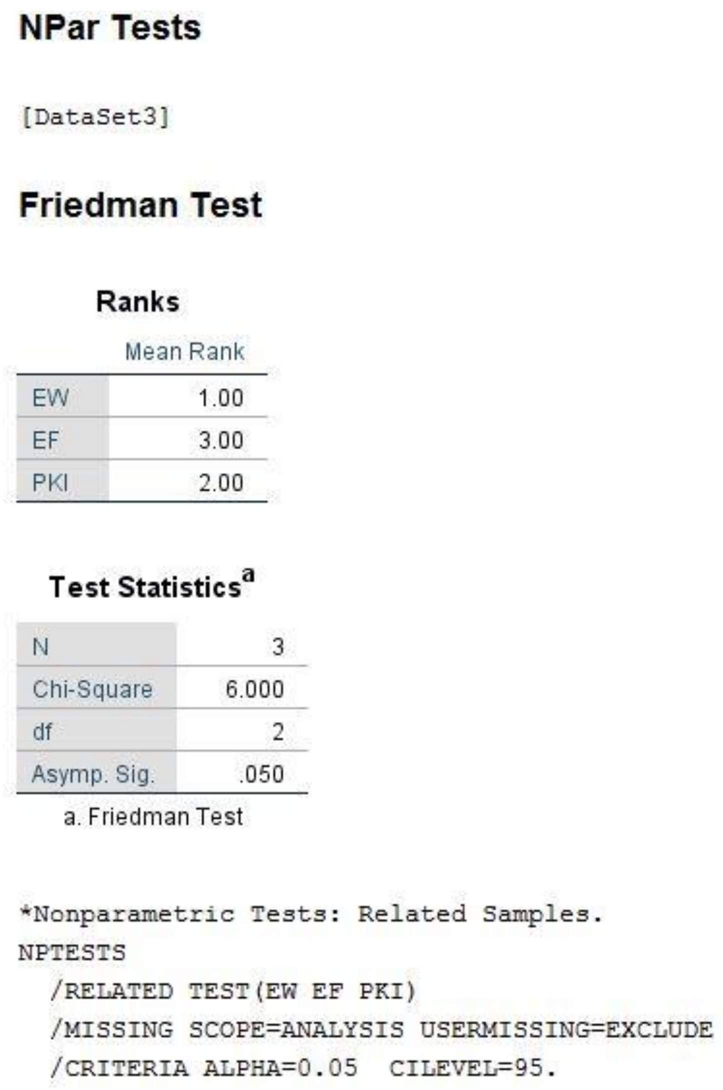
	Mean Rank
VAR00001	1.00
VAR00002	2.00
VAR00003	4.50
VAR00004	3.67
VAR00005	5.67
VAR00006	6.17
VAR00007	5.00
VAR00008	8.33
VAR00009	8.67
VAR00010	10.00

Test Statistics[a]

N	3
Chi-Square	24.988
df	9
Asymp. Sig.	.003

a. Friedman Test

Appendix D



NPar Tests

```
[DataSet3]
```

Friedman Test

Ranks

	Mean Rank
EW	1.00
EF	3.00
PKI	2.00

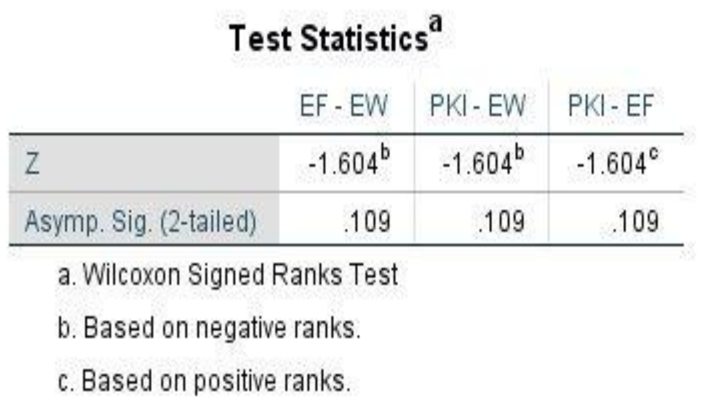
Test Statistics[a]

N	3
Chi-Square	6.000
df	2
Asymp. Sig.	.050

a. Friedman Test

```
*Nonparametric Tests: Related Samples.
NPTESTS
  /RELATED TEST(EW EF PKI)
  /MISSING SCOPE=ANALYSIS USERMISSING=EXCLUDE
  /CRITERIA ALPHA=0.05  CILEVEL=95.
```

Wilcoxon Test



Test Statistics[a]

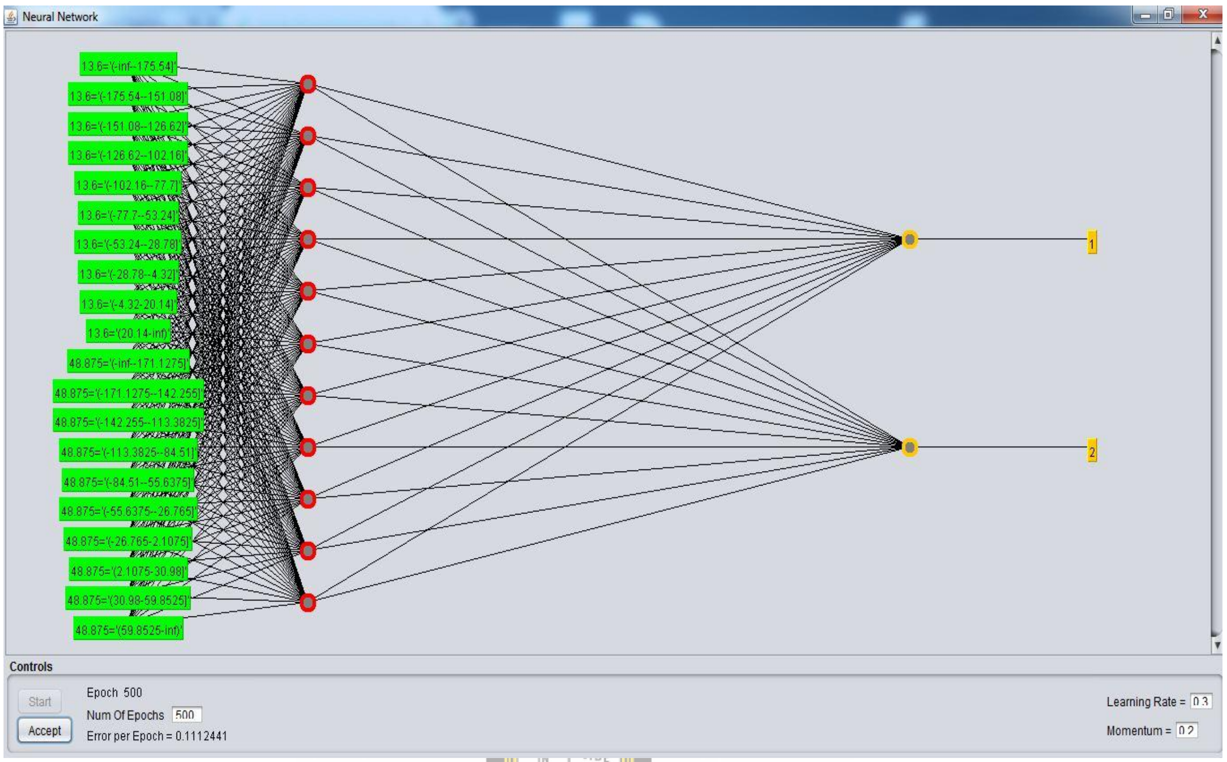
	EF - EW	PKI - EW	PKI - EF
Z	-1.604[b]	-1.604[b]	-1.604[c]
Asymp. Sig. (2-tailed)	.109	.109	.109

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

c. Based on positive ranks.

Appendix E – Artificial Neural Network Training



Epoch 500
Num Of Epochs 500
Error per Epoch = 0.1112441
Learning Rate = 0.3
Momentum = 0.2

```
=== Classifier model (full training set) ===

Sigmoid Node 0
    Inputs    Weights
    Threshold    14.663285519719123
    Node 2    -11.490529123210765
    Node 3    -14.858885027539177
Sigmoid Node 1
    Inputs    Weights
    Threshold    -14.66394230568059
    Node 2    11.488819825146841
    Node 3    14.860994224257
Sigmoid Node 2
    Inputs    Weights
    Threshold    -84.6887960764195
    Attrib 13.6    48.31968014772785
    Attrib 48.875    68.52603650850965
Sigmoid Node 3
    Inputs    Weights
    Threshold    -100.66451554324516
    Attrib 13.6    60.691419021252045
    Attrib 48.875    78.9865323216184
Class 1
    Input
    Node 0
Class 2
    Input
    Node 1
```