SITUATION RECOGNITION USING SOFT COMPUTING TECHNIQUES

By:

Pheeha MACHAKA

Supervisor:

Dr. Antoine BAGULA

THESIS PRESENTED FOR THE DEGREE OF MASTER OF SCIENCE

IN THE DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF CAPE TOWN

- JANUARY 2012 -

Plagiarism Declaration

I know the meaning of plagiarism and declare that all work in this document, save for that which is properly acknowledged, is my own.

Pheeha Machaka.

Dedications

To my sister, Sejabaledi Machaka

To my mother, Makoma Elizabeth Machaka

To my father, Matome John Machaka

I love you.

Acknowledgements

To God Almighty! The creator of heavens and the earth! The King of Kings! Jesus Christ!

I thank you for all the life privileges and opportunities. I thank you for keeping me safe, Lord.

The God Who Never Changes and Whose Love Never Fails!

To my Family for the continued love and support throughout my life.

To my Supervisor, Dr Antoine Bagula: I thank you and truly appreciate the support and guidance throughout this research. Thank you for your nurture, knowledge and experience imparted to me.

To my friends for helping and encouraging me: you all are amazing!

For financial assistance from the National Research Foundation (NRF) and the Telkom Centre of Excellence (CoE): thank you.

Publications

1. Machaka, P and Bagula, A (2011). Pre-emptive Performance Monitoring of a Large Network of Wi-Fi Hotspots: An Artificial Immune System. In Proceedings of the 9th International Conference on Wired/Wireless Internet Communications, Lecture Notes in Computer Science (LNCS), 2011, Volume 6649/2011, pages 494-504,June 15-17 2011, Vilanova i la Geltrú, Spain. DOI: 10.1007/978-3-642-21560-5_41.

2. Machaka, P, Bagula, A and de Wet, N (2012). A Highly Scalable Monitoring Tool for Wi-Fi Networks. In Proceedings of the 2012 IEEE 1st International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems. September 20-21 2012, Offenburg, Germany.

3. Machaka, P., Mabande, T. and Bagula, A. (2011). Monitoring of a Large Wi-Fi Hotspots Network: Performance Investigation of Soft Computing Techniques. In Proceedings of the 6th International ICST Conference on Bio-Inspired Models of Network, Information and Computing Systems, Bionetics 2011, Lecture Notes of the Institute for Computer Science, Social Informatics and Telecommunication Engineering (LNICST). December 5-7 2011, York, England.

4. Mabande, T., Bagula, A., Balikuddembe, J. and Machaka P. (2011). A Comparative Evaluation of Business Intelligence Technologies with Application to Product Profiling. In Proceedings of the 6th International ICST Conference on Bio-Inspired Models of Network, Information and Computing Systems, Bionetics 2011, Lecture Notes of the Institute for Computer Science, Social Informatics and Telecommunication Engineering (LNICST). December 5-7 2011, York, England.

Abstract

The last decades have witnessed the emergence of a large number of devices pervasively launched into our daily lives as systems producing and collecting data from a variety of information sources to provide different services to different users via a variety of applications. These include infrastructure management, business process monitoring, crisis management and many other system-monitoring activities. Being processed in real-time, these information production/collection activities raise an interest for live performance monitoring, analysis and reporting, and call for data-mining methods in the recognition, prediction, reasoning and controlling of the performance of these systems by controlling changes in the system and/or deviations from normal operation.

In recent years, soft computing methods and algorithms have been applied to data mining to identify patterns and provide new insight into data. This thesis revisits the issue of situation recognition for systems producing massive datasets by assessing the relevance of using soft computing techniques for finding hidden pattern in these systems. Building upon a case study methodology, the thesis work evaluates and compares the performance of three of the most known soft computing techniques, namely Artificial Immune Systems, Bayesian Belief Networks and Neural Networks when applied to different real-life datasets. These include a Wi-Fi network monitoring dataset collected from a wireless Internet Service Provider (ISP) in Cape Town, South Africa and a drought monitoring dataset taken from the Trompsburg area in Free State province, South Africa. The experimental results reveal the relevance of using these techniques in situation recognition as well as the strength and weaknesses of the algorithms behind each of these three soft computing techniques.

Keywords: Situation Recognition, Soft Computing, Data Mining, Artificial Neural Networks, Artificial Immune Systems, Bayesian Networks, Network Monitoring, Drought Monitoring.

Table of Contents

SITUATION RECOGNITION USING SOFT COMPUTING TECHNIQUES 1
1. INTRODUCTION 1
1.1. Background Information 1
1.2. Research Objectives and Research Questions 4
1.3. Contribution 4
1.4. Dissertation Outline 5
2. BACKGROUND AND RELATED WORK 7
2.1. Artificial Neural Networks 7
2.2. Bayesian Networks 11
2.3. Artificial Immune Systems 13
2.3.1. Artificial Immune System (AIS) and Negative Selection 13
2.3.2. Negative Selection Mechanism 14
2.3.3. Negative Selection Algorithm 14
2.4. Summary 16
3. RESEARCH DESIGN 17
3.1. Research Method 17
3.1.1. Case Studies 17
3.1.2. Experimentation 18
3.2. Sample Data 19
3.2.1. Challenges and Remedies of using sample data. 20
3.3. Domain Knowledge 21
3.4. Performance Evaluation of Machine Learning Algorithms 22
3.4.1. Measures of performance 22
3.4.2. Graphical Descriptive Techniques 25
3.4.3. Evaluating Machine Learning Algorithms' Performance on the same Dataset 25
3.5. Software Program and Packages 26
3.6. Algorithms Parameter Settings in Weka 27
3.6.1. Multilayer Perceptron 27
3.6.2. Bayesian Belief Networks 29
3.6.3. Artificial Immune Systems 29
4. REDBUTTON WI-FI NETWORK MONITORING CASE STUDY 31
4.1. Introduction 31

4.2. Background and Related Work 32
4.3. The Experimental Model 34
4.3.1. The Wi-Fi Network 34
4.3.2. Data Collection Methods 35
4.3.3. Performance Metrics 40
4.3.4. Experimental Design 41
4.3.5. Encoding of Performance Metrics 43
4.3.6. Test Cases 44
4.4. Experiment Results 45
4.4.1. Anomaly Performance 45
4.4.2. Aberrant Performance 55
4.4.3. Data Mining Using a Confusion Matrix 65
4.5. Conclusion 71
4.5.1. Anomaly Network Performance Detection 71
4.5.2. Aberrant Network Performance Detection 72
5. REDBUTTON NETWORK INTRUSION DETECTION CASE STUDY 73
5.1. Introduction 73
5.2. Background and Related Work 73
5.3. Research Methods 74
5.4. Experiment Results 75
5.4.1. Anomaly Performance 75
5.4.2. Aberrant Performance 85
5.4.3. Data Mining Using a Confusion Matrix 95
5.5. Discussion 100
5.6. Conclusion 101
5.6.1. Anomaly Network Performance Detection with Network Intrusion 101
5.6.2. Aberrant Network Performance Detection with Intrusion 102
5.6.3. Comparing Situation Recognition Before and After Network Intrusion 102
6. TROMPSBURG DROUGHT MONITORING CASE STUDY 105
6.1. Introduction 105
6.2. Background 105
6.2.1. Definitions of drought and its classifications 105
6.2.2. Drought Indices 106

6.3. Standard Precipitation Index (SPI) 108
6.3.1. Background of SPI 108
6.3.2. Calculating SPI 108
6.4. Related Work 110
6.5. Research Methods 111
6.5.1. Drought Monitoring Region and Data Collection 111
6.5.2. Test Cases 111
6.5.3. Experiment Design 111
6.6. Experiment Results 112
6.6.1. Kappa Statistic Performance 113
6.6.2. True Positive Rate Performance 114
6.6.3. False Positive Rate Performance 115
6.6.4. Precision Performance 117
6.6.5. Recall Performance 118
6.6.6. F-Measure Performance 120
6.6.7. ROC Area 121
6.6.8. Confusion Matrix 122
6.7. Discussion 126
6.8. Conclusion 126
7. CONCLUSION 128
8. BIBLIOGRAPHY 130

LIST OF ABBREVIATIONS

AI: Artificial Intelligence 3
AIRS: Atrificial Immune Recognition System 45
AIS: Artificial Immune Systems 4
ANN: Artificial Neural Networks 4
AP: Access Point 31
API: Application Programming Interface 101
ARB: Artificial Recognition Ball 30
ARIMA: Autoregressive Integrated Moving Average 16
AUC: Area Under Curve (For ROC - Receiver Operating Curve) 24
BBN: Bayesian Belief Networks 7
BN: Bayesian Networks 4
DAG: Directed Acyclic Graph 11
DARPA: Defence Advanced Research Project Agency 11
DNS: Domain Name Server 33
DoS: Denial of Service 39
EDI:Effective Drought Index 125
FN: False Negative 22
FP: False Positive 22
GeNIe: Graphical Network Interface 74
GPS: Global Positioning System 41
GUI: Graphical User Interface 27
IDS: Intrusion Detection Systems 11
IEEE: Institute of Electrical and Electronics Engineers 3
IP: Internet Protocol 38
ISM: Industrial, Scientific and Medical (Radio Band) 41
IVHS:intelligent Vehicle and Highway Systems 13
KDD99: Knowledge Discovery and Data Mining 1999 33
LAN: Local Area Network 39
MCDM: Multi-Criteria Decision Making 24
MIB: Management Information Base 35
MLP: Multilayer Perceptron 9
NSA: Negative Selection Algorithm 14
PDA: Personal Digital Assistant 31
PDSI: Palmer Drought Severity Index 106
PDU: Protocol Data Unit 37
QoS: Quality of Service 44
RFID: Radio Frequency Identification 41
ROC: Receiver Operating Characteristics 24
SARIMA: Seasonal Autoregressive Integrated Moving Average 16
SASCO: System for Automated Customer Support Operations 13
SLA: Service Level Agreement 31
SMILE: Structural Modelling, Inference, and Learning Engine 74
SNMP: Simple Network Managment Protocol 35
SPI: Standard Precipitation Index 4
SSL: Secure Socket Layer 39
TBP-NN: Temporal Back-Propagation Neural Network 110

TLS: Transport Layer Security 39
TN: True Negative 22
TP: True Positive 22
UDP: User Datagram Protocol 11
WEKA: Waikato Environment for Knowledge Analysis 27
Wi-Fi : Wireless Fidelity 3
WSN: Wireless Sensor Networks 1

LIST OF TABLES

Table 3.1: A Typical Confusion Matrix 25
Table 4.1 Parameter Setting for Each Experiment Test Case 44
Table 4.2:Non-self Detectors for the Experiment 45
Table 4.3: Results for Anomaly Kappa Statistic T-test for Paired Two Samples for Means 46
Table 4.4: Results for Anomaly True Positive Rate T-test for Paired Two Samples for Means 48
Table 4.5: Results for Anomaly False Positive Rate T-test for Paired Two Samples for Means 49
Table 4.6: Results for Anomaly Precision T-test for Paired Two Samples for Means 51
Table 4.7: Results for Anomaly Recall T-test for Paired Two Samples for Means 52
Table 4.8: Results for Anomaly F-Measure T-test for Paired Two Samples for Means 53
Table 4.9: Results for Anomaly ROC Area T-test for Paired Two Samples for Means 55
Table 4.10: Results for Aberrant Kappa Statistic T-test for Paired Two Samples for Means 56
Table 4.11: Results for Aberrant True Positive Rate T-test for Paired Two Samples for Means 58
Table 4.12: Results for Aberrant False Positive Rate T-test for Paired Two Samples for Means 59
Table 4.13: Results for Aberrant Precision T-test for Paired Two Samples for Means 61
Table 4.14: Results for Aberrant Recall T-test for Paired Two Samples for Means 62
Table 4.15: Results for Aberrant F-measure T-test for Paired Two Samples for Means 63
Table 4.16: Results for Aberrant ROC Area T-test for Paired Two Samples for Means 65
Table 5.1: Results for Anomaly Kappa Statistic T-test for Paired Two Samples for Means 76
Table 5.2: Results for Anomaly True Positive Rate T-test for Paired Two Samples for Means 78
Table 5.3: Results for Anomaly False Positive Rate T-test for Paired Two Samples for Means 79
Table 5.4: Results for Anomaly Precision T-test for Paired Two Samples for Means 80
Table 5.5: Results for Anomaly Recall T-test for Paired Two Samples for Means 82
Table 5.6: Results for Anomaly F-Measure T-test for Paired Two Samples for Means 83
Table 5.7: Results for Anomaly ROC Area T-test for Paired Two Samples for Means 84
Table 5.8: Results for Aberrant Kappa Statistic T-test for Paired Two Samples for Means 86
Table 5.9: Results for Aberrant True Positive Rate T-test for Paired Two Samples for Means 88
Table 5.10: Results for Aberrant False Positive Rate T-test for Paired Two Samples for Means 89
Table 5.11: Results for Aberrant Precision T-test for Paired Two Samples for Means 90
Table 5.12: Results for Aberrant Recall T-test for Paired Two Samples for Means 92
Table 5.13: Results for Aberrant F-measure T-test for Paired Two Samples for Means 93
Table 5.14: Results for Aberrant ROC Area T-test for Paired Two Samples for Means 94
Table 6.1: SPI Classification and Corresponding Probabilities of Occurrence [56, 57] 108
Table 6.3: Results for Kappa Statistic T-test for Paired Two Samples for Means 113
Table 6.4: Results for True Positive Rate T-test for Paired Two Samples for Means 114
Table 6.5: Results for False Positive Rate T-test for Paired Two Samples for Means 116
Table 6.6: Results for Precision T-test for Paired Two Samples for Means 117
Table 6.7:Results for Recall T-test for Paired Two Samples for Means 119
Table 6.8: Results for F-Measure T-test for Paired Two Samples for Means 120
Table 6.9: Results for ROC Area T-test for Paired Two Samples for Means 122
Table 6.10: Multilayer Perceptron Confusion Matrices for (a) SPI3 and (b) SPI6 Test Cases 123
Table 6.11: Multilayer Perceptron Confusion Matrices for (c) SPI12 and (d) SPI24 Test Cases 123
Table 6.12: NaiveBayes Confusion Matrices for (a) SPI3 and (b) SPI6 Test Cases 124
Table 6.13: Multilayer Perceptron Confusion Matrices for (c) SPI12 and (d) SPI24 Test Cases 124
Table 6.14: AIRS2 Confusion Matrices for (a) SPI3 and (b) SPI6 Test Cases 125
Table 6.15: AIRS2 Confusion Matrices for (c) SPI12 and (d) SPI24 Test Cases 125

LIST OF FIGURES

Figure 1.1: Situation Management Domain (adapted from [43]) 2
Figure 1.2: Machine Learning Techniques and Case study Application 4
Figure 1.3: Dissertation Outline 6
Figure 2.1: A schematic drawing of the Biological Neuron [33] 8
Figure 2.2: The Artificial Neuron 9
Figure 2.3: A diagram depicting a typical Multi-Layered Perceptron Model 10
Figure 2.4: A Bayesian Network for the Family-Out Problem [10] 12
Figure 2.5: A pictorial representation of the acquired immune system mechanism (taken from [17]) 13
Figure 2.6: (a) Generation of valid detector set (stage 1); (b) Monitoring the protected strings for changes (stage 2). (Adapted from [23]) 14
Figure 3.1: Sample Data split into Training and Testing set. 19
Figure 3.2: A typical Precision-Recall Graph 24
Figure 3.3: ROC curves for two machine learning algorithms (taken from [91]) 24
Figure 4.1: Graph showing worldwide public hotspots per location 32
Figure 4.2: The Wi-Fi Network Monitoring Tool 34
Figure 4.3: Centralised Client-Server Model 35
Figure 4.4: Experimental Plan 42
Figure 4.5: Anomaly Kappa Statistic Performance Bar Chart 46
Figure 4.6: Anomaly True Positive Rate Performance Bar Chart 47
Figure 4.7: Anomaly False Positive Rate Performance Bar Chart 49
Figure 4.8: Anomaly Precision Performance Bar Chart 50
Figure 4.9: Anomaly Recall Performance Bar Chart 52
Figure 4.10: Anomaly F-measure Performance Bar Chart 53
Figure 4.11: Anomaly ROC Area Performance Bar Chart 54
Figure 4.12: Aberrant Kappa Statistic Performance Bar Chart 56
Figure 4.13: Aberrant True Positive Rate Performance Bar Chart 57
Figure 4.14: Aberrant False Positive Rate Performance Bar Chart 59
Figure 4.15: Aberrant Precision Performance Bar Chart 60
Figure 4.16: Aberrant Recall Performance Bar Chart 62
Figure 4.17: Aberrant F-measure Performance 63
Figure 4.18: Aberrant ROC Area Performance 64
Figure 4.19: Multilayer Perceptron Anomaly Detection Bar Chart 66
Figure 4.20: NaiveBayes Anomaly Detection Bar Chart 67
Figure 4.21: AIRS2 Anomaly Detection Bar Chart 68
Figure 4.22: MLP Aberrant Detection Bar Chart 69
Figure 4.23: NaiveBayes Aberrant Detection Bar Chart 70
Figure 4.24: AIRS2 Aberrant Detection Bar Chart 70
Figure 5.1: Anomaly Kappa Statistic Performance Bar Chart 76
Figure 5.2: Anomaly True Positive Rate Performance Bar Chart 77
Figure 5.3: Anomaly False Positive Rate Performance Bar Chart 79
Figure 5.4: Anomaly Precision Performance Bar Chart 80
Figure 5.5: Anomaly Recall Performance Bar Chart 81
Figure 5.6: Anomaly F-measure Performance Bar Chart 83
Figure 5.7: Anomaly ROC Area Performance Bar Chart 84
Figure 5.8: Aberrant Kappa Statistic Performance Bar Chart 86

Figure 5.9: Aberrant TP Rate Performance Bar Chart 87
Figure 5.10: Aberrant False Positive Rate Performance Bar Chart 89
Figure 5.11: Aberrant Precision Bar Chart 90
Figure 5.12: Aberrant Recall Performance Bar Chart 91
Figure 5.13: Aberrant F-measure Performance Bar Chart 93
Figure 5.14: Aberrant ROC Area Performance Bar Chart 94
Figure 5.15: Multilayer Perceptron Anomaly Network Performance Detection Bar Chart 96
Figure 5.16: NaiveBayes Anomaly Network Performance Detection Bar Chart 97
Figure 5.17: AIRS2 Anomaly Network Performance Detection Bar Chart 97
Figure 5.18: MLP Aberrant Network Behavior Detection Bar Chart 98
Figure 5.19: NaiveBayes Aberrant Network Behavior Detection Bar Chart 99
Figure 5.20: AIRS2 Aberrant Network Behavior Detection Bar Chart 99
Figure 6.1: Trompsburg Experiment 111
Figure 6.3: SPI Kappa Statistic Performance Bar Chart 113
Figure 6.4: SPI True Positive Rate Performance Bar Chart 114
Figure 6.5: SPI False Positive Rate Performance Bar Chart 115
Figure 6.6: SPI Precision Performance Bar Chart 117
Figure 6.7: SPI Recall Performance Bar Chart 118
Figure 6.8: SPI F-Measure Performance Bar Chart 120
Figure 6.9: SPI ROC Area Performance Bar Chart 121

1. INTRODUCTION

1.1. Background Information

The last decades have witnessed the emergence of large number of devices, both fixed and mobile, pervasively launched into our daily lives to produce and collect data from a variety of information sources. This data production/collection is used to provide different services to different users in a heterogeneous environment involving different applications from different fields such as Financial Investments, Health Care, Telecommunications and the World Wide Web. These applications are characterised by an influx of large volumes of data. For example, according to the World Internet Statistics website, from year 2000-2011 the population of internet users has grown by 706.9% in Asia alone, while in Africa this population of internet users grew by 2 527.4% between years 2000 to 2011 [41]. A major contributing factor to this growth may have been the growth and use of social media like Twitter and Facebook.

According to [55], social websites released statistics relating to activity on social websites showing there has been great growth since 2007. Twitter states that in January 2010 they were experiencing 50 million tweets a day, which rounds off to about 600 tweets per second.

Facebook also released statistics saying their users post more than 60 million status updates per day, which rounds off to about 700 status updates per second [55].

Google search engine in early 2011 released statistics that says it receives 121 million searches per hour, which are about 2 million searches per minute and about 34 000 searches per second [55].

These are massive amounts of data that is relevant to companies' marketing, product research and developments. The data pertains to information about their current and potential customers' online activities. The data can reveal information that answers questions like: what are the customers saying about the product? How do they perceive the brand? And many other questions that refer to the full customer experience. The World Wide Web is not the only domain that is characterised by influxes of large volumes of data.

A very important and critical example is in disaster management. Natural and man-made disasters such as earthquakes, floods, droughts and forest fires pose a challenge to public emergency services. In order for stakeholders to respond in a fast and well-coordinated manner, information concerning the situation plays a vital role. Stakeholders involved in the response may include the police department, health department, department of defence and fire department. The information communicated among the departments needs to be captured, centralised and well-coordinated.

Wireless Sensor Networks (WSN) are promising a revolution in the way critical information is captured, processed and exchanged [21]. WSNs are made of spatially distributed autonomous sensor nodes that are designed to monitor environment conditions and changes.

In a typical deployment scenario, sensor nodes can monitor variables like temperature, humidity, pressure and pollution. The data variables collected are then collectively sent to a central node which has higher computational power for further processing. Data are stored in central databases that allow large-scale storage, manipulation and access to data. The data can then be translated into information that is useful to the stakeholders and thus allowing and enabling response to disastrous situations within a predefined operational time frame. This deployment scenario requires methods for situation monitoring, situation awareness and situation control. According to [42], the critical aspects of the scenario include "managing and controlling sources of information, processing real-time or near real-time streams of events, representing and integrating low-level events and higher-level concepts, multi-source information fusion, information presentation that maximises human comprehension, reasoning about what is happening and what is important". This whole process can be termed Situation Management.

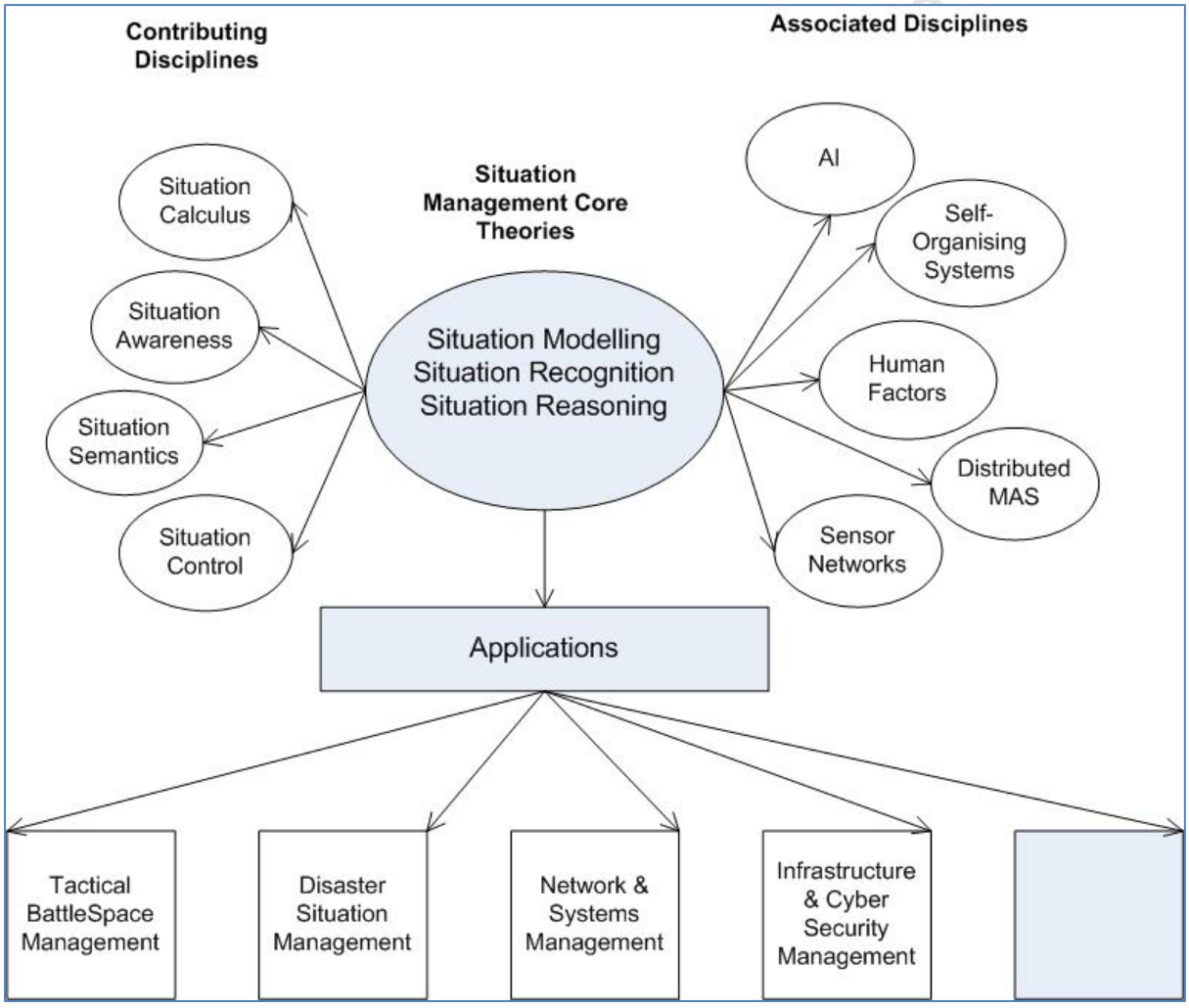

Figure 1.1: Situation Management Domain (adapted from [43])

In [43], Situation Management is defined as a "synergistic goal-directed process of

a) sensing and information collection,
b) perceiving and recognising process,

c) analysing past situations and predicting future situations, and
d) reasoning, planning and implementing actions so that desired goal situation is reached within some pre-defined constraints".

The research community in situation management is fairly young as it was launched in 2005 at its first Workshop on Situation Management (SIMA 2005) in association with the IEEE Communications Society Military Communications Conference (MIL-COM).

This has led to a growing interest among researchers for situation management, and to contributions to this field of research. These contributions include the work by Jakobson *et al*. in [43], were an overall "big picture" of situation management is given as represented by Figure 1.1. Figure 1.1 above shows the core theories of situation management, its contributing disciplines, its associated disciplines and the applications of situation management. Core research theories of situation management include Situation Modelling, Situation Recognition, and Situation Reasoning. While explaining the research field of situation management, Jakobson included other associated stand-alone disciplines such as Artificial Intelligence (AI), Information Fusion, Distributed Multi-Agent Systems, Semantic Web, Sensor Networks, Self-organising Systems and Human Behaviour. Applications of Situation Management are varied and inter-disciplinary. Just to mention a few, its applications include Tactical Battlespace Management, Disaster Situation Management, Networks & Systems Management, and Infrastructure & Cyber Security Management.

The research focus of this thesis work lies in Situation Recognition using AI techniques. Situation Recognition is the branch of situation management that is concerned with collection and interpretation of data. This involves the discipline of data mining, where artificial intelligence techniques have been applied.

Three of the most widely used AI techniques include the Artificial Immune System and the Artificial Neural Networks and Bayesian Networks. While these techniques have gained success and have been widely used in data mining and soft computing, it is not well known which one of them performs better than the other and how do the three techniques perform when compared against each other. The data-mining efficiency of these techniques when applied to different datasets is also a topic that has been only poorly addressed by the research community. These issues are addressed in this thesis work with the attempt to shed light on the relevance of using these techniques in real-life applications.

1.2. Research Objectives and Research Questions

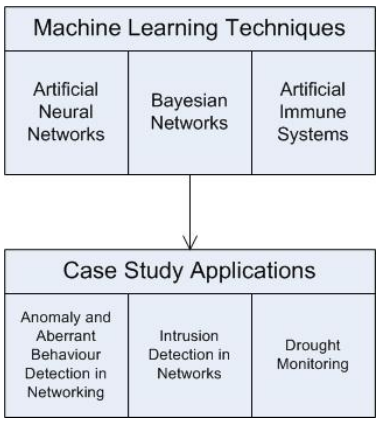

Figure 1.2: Machine Learning Techniques and Case study Application

The figure 1.2 indicates a hierarchal outline of the three machine learning techniques used in the thesis, and the three case study application. The research investigation made out for this thesis was carried out as part of a middleware research work for two research projects. These include a research project aiming at developing a monitoring tool for a large scale Wi-Fi network with the objective of

- determining which intelligent algorithm performs better for network monitoring and intrusion detection
- evaluating how the algorithm performs under different test cases and network thresholds
- evaluating how the algorithms perform for anomaly and aberrant behaviour detection
- investigating what kind of information can be mined using the intelligent algorithms.

The second research project aimed to achieve drought monitoring using datasets collected from off-the-shelf sensor devices and professional weather stations located in cities by public administrations. As part of the middleware project, we wanted to:

- determine which intelligent algorithm performs better for the drought-monitoring case study
- evaluate how the intelligent algorithms perform across different Standard Precipitation Index (SPI) time scales
- Investigate what kind of information can be mined using the intelligent algorithms.

1.3. Contribution

In the research community there has been research or scholar work that evaluates the performance of the Artificial Immune Systems, Bayesian Networks and Artificial Immune Systems, individually. These evaluations and investigations have been applied to various problem domains, and mostly in network monitoring and intrusion detection. While there

have been attempts to compare and evaluate the performance of the three machine learning techniques, to the best of our knowledge at the time of writing this thesis work, there has not been work that jointly compare the performance of the AIS, BN and ANN algorithms in the context of network monitoring, network intrusion detection and drought monitoring.

The key theoretical and practical contributions to the research and knowledge of this thesis work consist of:

- determining which computational intelligence technique is better suited for Wi-Fi network monitoring, network intrusion detection and drought monitoring
- providing a machine learning technique that can be used in developing intelligent middleware for the context of network monitoring and drought monitoring, and
- identifying performance parameters and providing a combination of novel designs and experimentation methods which other researchers can use and build upon for future research.

1.4. Dissertation Outline

The structure and outline of the dissertation is as shown by the Figure 1.3. Chapter 2 of the dissertation will discuss the background and related scholar research work in soft computing and computational intelligence. The chapter will give a comprehensive and contextual literature review of the artificial immune systems, Bayesian networks and artificial neural networks. The chapter will also look into scholar work that compares performance of the three algorithms, which is the aim of this dissertation, in a context of the three case studies presented.

Chapter 3 of the dissertation will give a description of research methods used to carry out the research investigations. This includes data-collection methods, experiment methods and evaluation methods. The advantages, disadvantages and limitations of the methods are also discussed.

Chapter 4-6 are the three cases study investigations carried out for the dissertation. These chapters will give a comprehensive and contextual background and scholar work of each case study investigation; the experiment designs are discussed together with the experiment results and sub-conclusions.

Chapter 7 of the dissertation will discuss and summarise findings for the case study chapters 4-6, and give a conclusion about the research work. The future direction of the research work is also discussed in the final chapter.

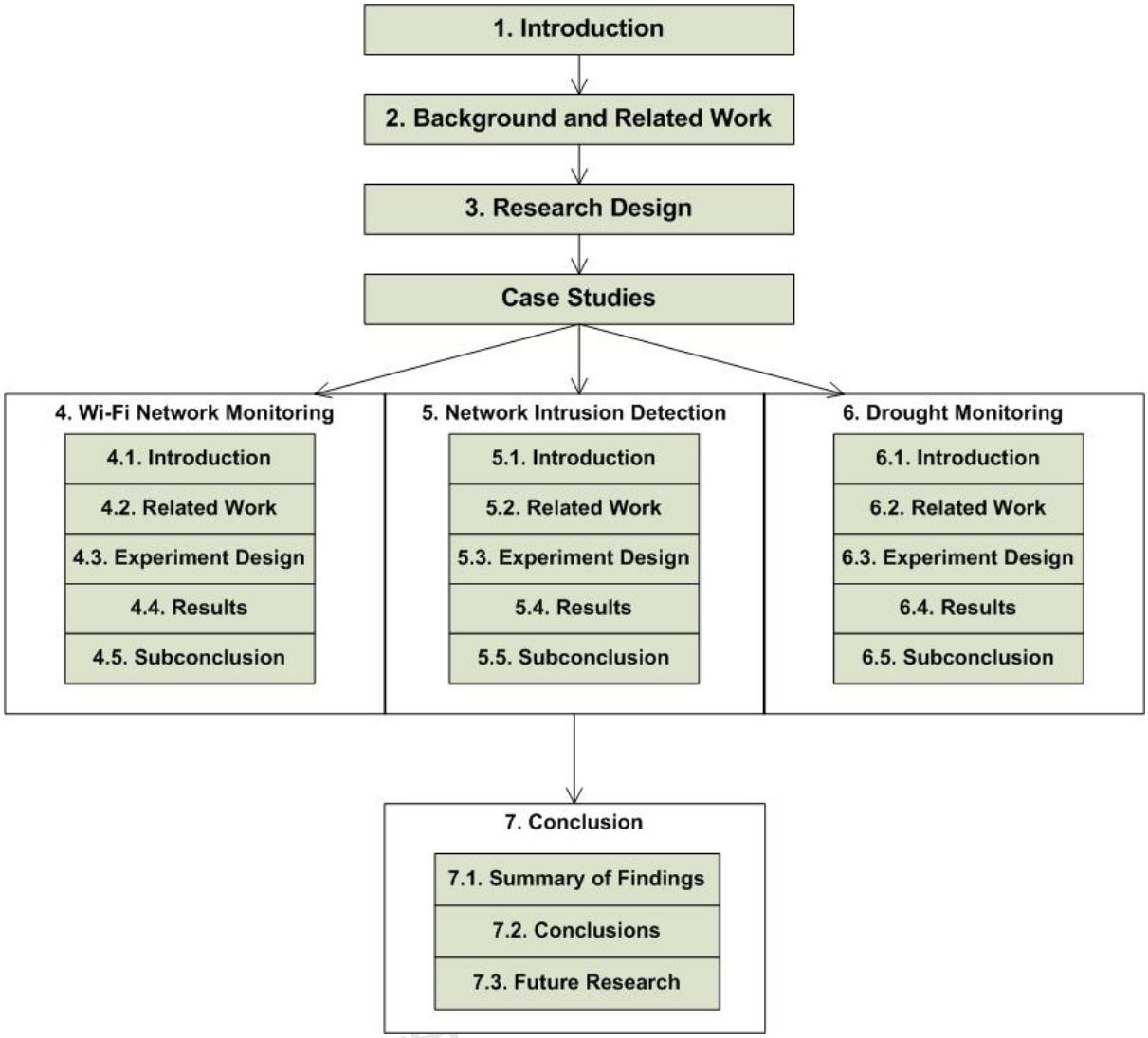

Figure 1.3: Dissertation Outline

2. BACKGROUND AND RELATED WORK

We are witnessing the emergence of a large number of data-producing/collecting devices deployed in our daily lives to provide different services to different applications used in different fields such as infrastructure monitoring, network monitoring, business process monitoring, crisis management, and other monitoring and management systems. Being collected in real-time, this information production/collection process raises an interest for live performance monitoring, reporting and analysis to monitor changes in the system, or deviations from normal performance. To monitor these changes and/or deviations, there is a need for methods that can help with recognising, predicting, reasoning and controlling the performance of the system. The activities can be collectively identified as situation management [42].

For situation management to be a success, the data collected from the different information sources need to be extracted, summarised and represented in a way that aid decision making. Data mining is a process of identifying new patterns and insight into data. In recent years, soft computing systems have been applied to the field of data mining and pattern recognition. These systems include Artificial Neural Networks, Fuzzy Systems, Evolutionary Algorithms, Artificial Immune Systems, Bayesian Belief Networks (BBN) and Probabilistic Reasoning, to name just a few [48] .

In this thesis work, we will look into three soft computing systems that are widely used and have gained great success in the past [15]**,** namely Artificial Immune System, Artificial Neural Networks and Bayesian Belief Networks (BBN) when applied to achieve situation recognition on two different systems: (a) a Wi-Fi performance monitoring system and (b) a Drought monitoring system. These systems and the underlying techniques are described in the sections that follow.

2.1. Artificial Neural Networks

The human brain is a highly interconnected network of about 10^{11} or 100 billion biological neurons. This network of biological neurons facilitates important vital human body functions like reading, breathing, motion and thinking. Some parts of the neural structures are present in the human body since birth, while others are established by experience [58].

All biological neural features, including memory, are stored in the neurons and in their connections between them. Learning is viewed as a new connection between the neurons or the modification of existing neurons. This continues to change throughout life [33].

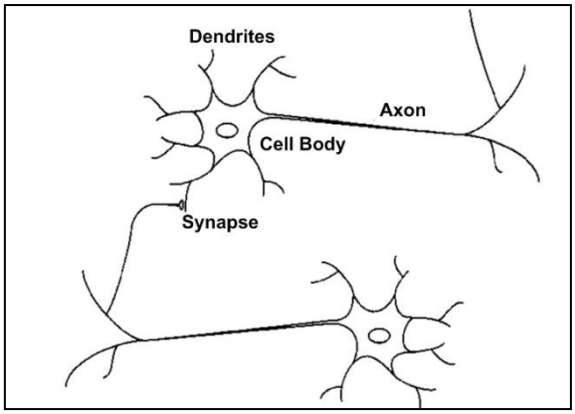

Figure 2.1: A schematic drawing of the Biological Neuron [33]

Figure 2.1 above represents a schematic drawing of the biological neuron. For the purposes of this research and for simplicity, the biological neuron is made up of three main components: the dendrite, the cell body, and the axon.

The dendrite is a tree-like network of nerve fibres that carry electrochemical signals from other neural cells, into the cell body of the dendrite. (The point of contact between the neurons is a synapse).

The cell body receives incoming electrochemical stimulation/signal from other dendrites and will effectively sum up and threshold these signals, and determines the extent to which action potentials are produced by the neuron.

The axon is a single long fibre of a nerve cell that carries electrochemical signals from the cell body out of the other neurons; and this is conducted by the synapse, which is a point of contact between axon of one cell and the dendrite of another cell, and sometimes contacts with muscles or gland cell.

The neural structure/network continues to develop throughout life and consists mainly of strengthening or weakening of synaptic junctions.

There has been significant research for understanding the functioning of the brain and biological neurons, and scientists have long had the desire to understand the brain and emulate its behaviour. This has allowed researchers to build mathematical models for the biological neurons such that the brain's working model can be simulated. Researchers therefore came up with the idea of the biologically inspired Artificial Neural Networks (ANN) [58, 71] .

ANNs (sometimes referred to as neural networks) are mathematical or computational models that get their inspiration from biological neural systems. They are information-processing systems with certain computational properties that are analogous to those that are learned from the biological neural network. The simplified biological neuron basically receives electrochemical signals from another neuron, for processing by allowing a specific action to be fired when a certain threshold is reached. In this way an artificial neuron can be seen as computational units that receive signals from other units [71] .

The ANN is made up of large networks of simple processing units called neurons or nodes along with their connections (synapse) in the network. The connection from one neuron to another has an associated weight, which encodes the knowledge of the network.

Figure 2.2 displays the structure of an artificial neuron. The nodes (or neurons) can be seen as computational units that receive input to process and produce an output. The processing might be simple (as in addition, subtraction, etc.) or it can be a complex computation (involving another network).

The processing is computed using a simple or complex mathematical function, the activation function. The activation function is computed using the input from other neurons (nodes) to give the output, and may involve a certain threshold depending on the problem domain.

The connection (synapse) between the neurons indicates an association between them (pointing from one neuron to the other). Each connection has a set of weights which encode the knowledge in the network.

ANNs have been used by scientists across a variety of research fields. This includes aerospace, banking, medicine and speech recognition, to mention just a few. Across most of the fields, different ANN models were used for problem solving as in such cases the problem can become complex. There exist several ANN architectures including fully connected networks, acyclic neural networks, modular neural networks and feed-forward neural networks. It is widely recognised that for complex tasks, ANN models with feed-forward multiple layers of neurons are necessary. This ANN model is generally referred to as the Multiple Layer Perceptron (or simply MLP) [20].

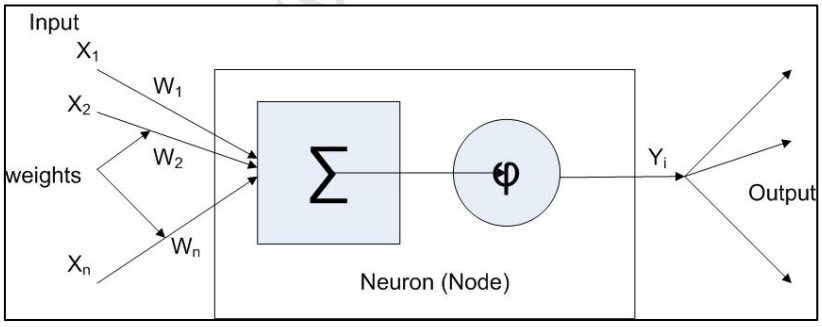

Figure 2.2: The Artificial Neuron

The essential structure of the MLP is that of perceptron (or nodes) interconnected in layers with the output from one layer forming the input to the next layer. Generally, the model is made up of three layers (this may differ from one textbook/author to the next): The input layer, hidden layer and the output layer. Figure 2.3 displays a representation of the generally used MLP model (three layers), where the output $Y_1 = f(x_1, x_2 \ldots x_n)$

The Input Layers - the nodes at this layer are input nodes; they encode and prepare the raw network data for further processing.

The Hidden Layers – this layer is not directly visible to the input and output layers. The nodes or neurons at this layer compute an internal representation of the data. With the use of MLP algorithms like the back propagation, the values of the connections (or weights) are adjusted accordingly, and algorithms may add non-linearity features, as opposed to linear best-fit algorithms that try to minimize the distance between the misclassified or the mispredicted.

The Output Layers – the neurons at this layer encode data representations from the results obtained by the computations at the hidden layer [20].

In [39], the authors address problems with the classification data mining techniques, and attempts to solve these problems using Neural Networks. The approach to data mining consisted of three phases:

- Constructing and training the network to correctly classify tulles in the given training data set to required accuracy;
- Pruning the network while maintaining the classification accuracy, and
- Extracting symbolic rules from the pruned network.

The experiment was carried out on a set of datasets and high quality rules were discovered using the proposed approach.

In [50] the authors propose a Genetic Algorithm approach to instance selection in artificial neural networks for financial data mining. The aim was to reduce the dimensionality if data and eliminate noisy and irrelevant instances. The proposed technique was applied on an 8 year (January 1991 to December 1998) technical data and the directions of change in the daily Korea Stock Price Index (KOSPI). The algorithm directly removes irrelevant and redundant instances from the training data, and thus improves forecasting and analysis for financial data.

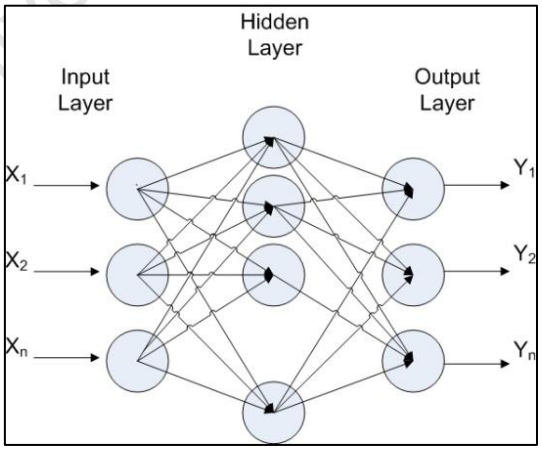

Figure 2.3: A diagram depicting a typical Multi-Layered Perceptron Model

In [6] the authors carried out experiments in the Bedup River in Sarawak, Malaysia. In their experiments they aimed at improving accurate water prediction without having precipitation data. The absent or missing precipitation data made it difficult for them to accurately predict

water levels. They used the Back propagation properties of Artificial Neural Networks to predict both the missing precipitation and water levels. The developed model was able to estimate the missing precipitation data with 96.4% accuracy, and the estimated data was used to further forecast the water levels with 85.3% accuracy compared to the 71.1% accuracy that was previously attained with the missing precipitation data. The artificial Neural Network techniques can be used in forecasting (prediction) of data, and therefore serves as a good tool for predication in situation management.

The work in [7, 8, 11, 92] indicates ANN being used for anomaly detection on network traffic data.
There has been work done in network intrusion detection using ANN. Authors in [7, 8] developed an intrusion detection systems that analysed UDP network traffic packets for misuse detection. The neural networks' MLP algorithm with backpropagation was used, and the results reveal a 98.2% training data correlation and a 97.5% test data correlation.
The authors in [11] have identified Artificial Immune Systems as very promising technique to intrusion detection. In their research they developed a system, Elman Networks, which uses the Dynamic Back Propagation algorithm. This was applied on the DARPA IDS (Defence Advanced Research Project Agency Intrusion Detection Systems) Evaluation dataset. The research results reveal that the Elman Networks system had was able to correctly classify 92.7% of the test instances.

2.2. Bayesian Networks

Since the invention of the computer, there has been a desire for the computer to perform intellectually challenging task. According to [47], solving an intellectually challenging task can be characterized as a process of deriving conclusions (new pieces of knowledge) by manipulating a (large) body of knowledge, typically including definitions of entities (objects, events, phenomena, etc.), relations among them, and observations of states(values) of some of the entities.

Systems that help with performing intellectually challenging task are those with artificial intelligence capabilities. One in particular is the Bayesian Networks Systems.

Bayesian Networks can be described briefly as acyclic directed graph (DAG) which defines a factorization of a joint probability distribution over the variables that are represented by the nodes of the DAG, where the factorization is given by the direct links of the DAG [47].

In the past decade, Bayesian Networks have become a popular representation for encoding uncertain expert knowledge in expert systems [37]. They are graphical tools or models for modelling causes and effect in many different domains. They are compact networks of probability that capture the probabilistic relationship between variables, as well as historical information about their relationships [66]. Inference and probabilities can be calculated using the Bayes Rule (formula) which states that: The probability of a hypothesis H (involving a set of variables) given some observed event (assignments of values to a set of evidence variables e), [37] .

$$P(H|e) = \frac{P(e|H).P(H)}{P(e)}$$

To Explain Bayesian Networks paradigm, the following popular example, the "family-out" problem, depicted by Figure 2.4 is used. In this example, the family member would describe the problem as follows:

Suppose when I go home at night, I want to know if my family is home before I try opening the door. Now, often when my wife leaves the house, she turns on an outdoor light. However, she sometimes turns on this light if she is expecting a guest. Also, we have a dog. When nobody is home, the dog is put in the backyard. The same is true if the dog has a bowel problem. Finally, if the dog is in the background, I will probably hear her barking.

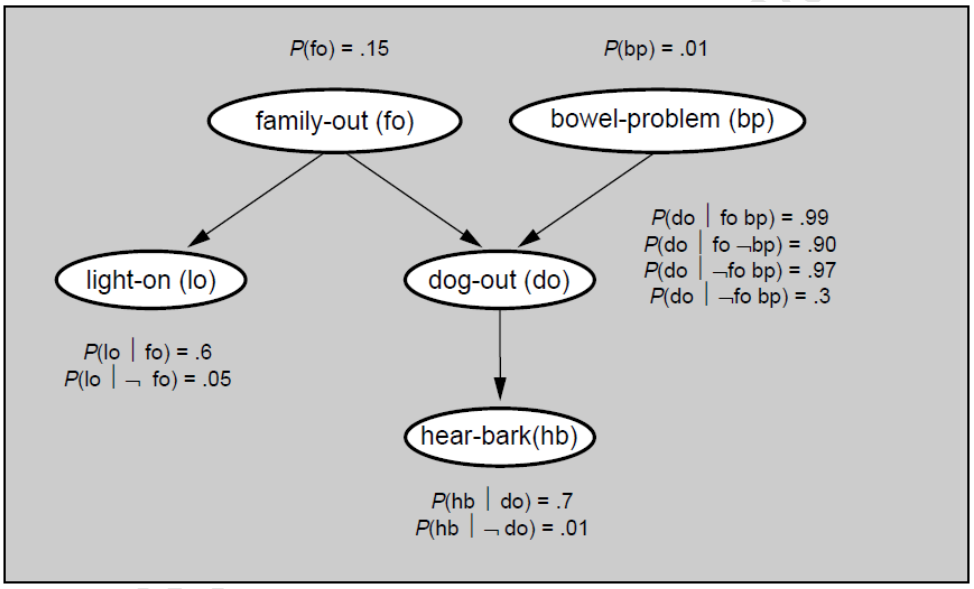

Figure 2.4: A Bayesian Network for the Family-Out Problem [10]

With Bayesian networks, one needs to give prior probabilities of all root nodes (nodes without predecessors) and the conditional probabilities of all non-root nodes given all possible combinations of their direct predecessors.

The Bayesian network depicted by Figure 2.4 depicts that 60% of the times, the light will be on if the family is out, but 5% of the times, the light will be on even when the family did not leave. In this way, Bayesian Networks provide a method to help perform reasoning and decision making under uncertainty and acquire knowledge from data/experience and to solve problems efficiently and respond to new situations.

In [2] the authors developed a Bayesian Belief Network, **HailFinder**, which combines meteorological data and BBN models to forecast severe weather in North-eastern Colorado. This was one of the first forecasting systems developed using BBN techniques.

The authors of [45]used Bayesian Belief Networks to develop the System for Automated Customer Support Operation (SASCO). This system was developed in partnership with Hewlett-Packard for trouble-shooting printing systems. Printers are complex systems with several components and trouble shooting is not a trivial task. The SASCO system automates this process.
In [22], the authors developed a system BATmobile for Intelligent Vehicles and Highway Systems (IVHS) which aims to substantially reduce congestion and accidents on the roads. They used Bayesian Networks for decision making given different conditions that were observed. The next section gives a description of the Artificial Immune Systems.

2.3. Artificial Immune Systems

The Human Immune System (HIS) is a robust, complex network of specialized cells and organs that defend the human body against a sea of harmful microorganisms called antigens. Building upon its capability of differentiating the cells and molecules of the body into self (what belongs) and non-self (what does not belong) [75] and [44], the HIS has led to the development of the Artificial Immune System (AIS) computational paradigm that may solve complex computational problems.

As depicted by Figure 2.5, once the body recognizes an invasion by antigens (I-III), the immunological response (IV-VI) consists of neutralizing or destroying the invading antigens through the release and use of B-cells and T-cells (lymphocytes that originate from the bone marrow and play an important role in the immunological response). The B and T-cells will then destroy the invading antigens and remove them from the body.

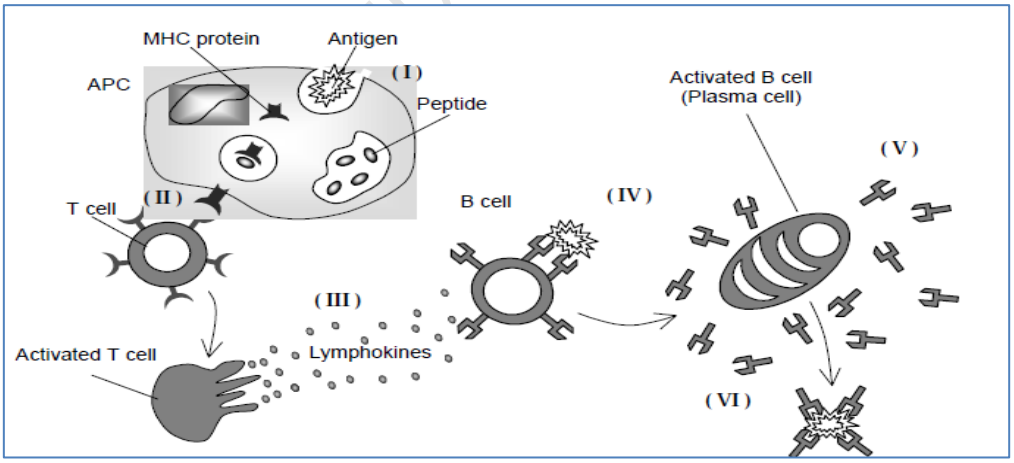

Figure 2.5: A pictorial representation of the acquired immune system mechanism (taken from [17])

2.3.1. Artificial Immune System (AIS) and Negative Selection

A biologically inspired computational technique, called Artificial Immune System (AIS) has emerged from the HIS metaphor. The HIS features that are of particularly relevant to AIS include matching, diversity and distributed control. These features have been implemented in

many AIS mechanisms such as the immune system's Immune Network Theory, Negative Selection mechanism and Clonal Selection Principle to solve real world science and engineering problems. The focus of this dissertation lies on the negative selection mechanism of the immune system with its application to pattern recognition in Wi-Fi network monitoring.

2.3.2. Negative Selection Mechanism

The HIS mechanism provides tolerance for self-cells, and reacts against unknown antigens. The HIS generates antibodies through a pseudo random rearrangement process and undergoes a censoring process in the thymus as described by [75] and [44]. Only those antibodies that do not bind to the self-protein are allowed to leave the thymus to circulate throughout the body to perform immunological functions and protect the body against foreign antigens. This mechanism gave developments to the Negative Selection Algorithm (NSA) described below.

2.3.3. Negative Selection Algorithm

In [23], the negative selection algorithm (NSA) was first proposed as a means of detecting unknown or illegal strings for virus detection in computer systems. This was inspired by a change detection mechanism which is based on the way the human immune system distinguishes self cells from non-self. The algorithm follows the two steps shown by Figure 2.6.

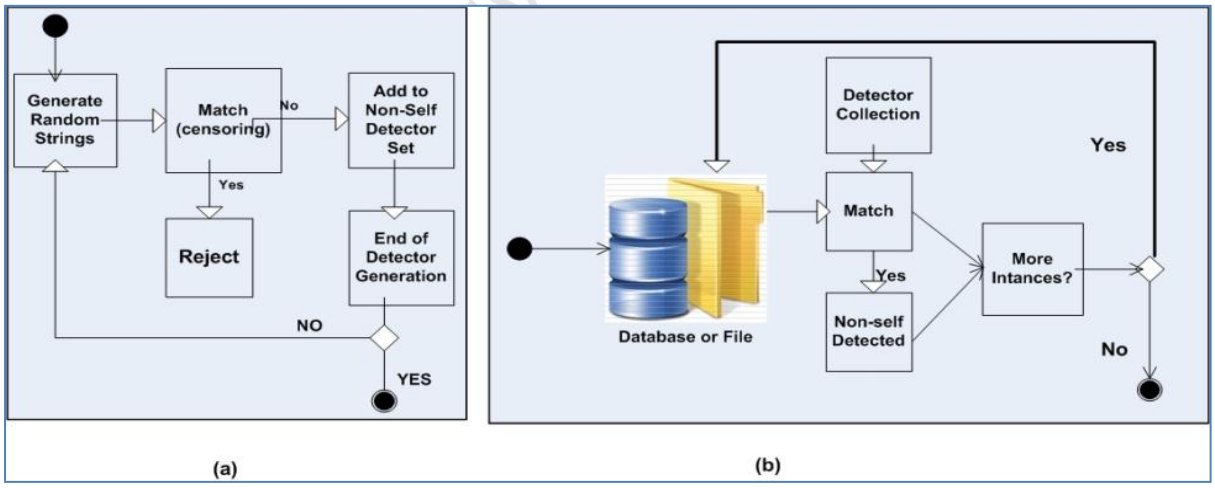

Figure 2.6: (a) Generation of valid detector set (stage 1); (b) Monitoring the protected strings for changes (stage 2). (Adapted from [23])

a) **Generate a set of detectors.** Each detector is a string representation that does not match any of the protected data (self). This is the censoring stage shown in Figure 2.6(a). The number of detectors generated will vary according to combination of total number of monitored variables and those that are undesired variables. If a non-self

detector is found, it is added to the detector set. This process is repeated until the desired number of detectors is reached or no more detectors are found.

b) **Monitor the protected strings.** The AIS continuously monitor for changes in the system, by matching encoded instances from the database/system. If a non-self detector is activated, the correct action will be taken, thus creating an alert and logging this with the monitoring system as show in Figure 2.6(b).

In [14, 24], the authors present a novel detection algorithm that is inspired by the negative selection mechanism of the immune system. In this algorithm, there is a distinction between the *self* and the *non-self*. The algorithms were applied to a simulated cutting dynamics in a milling operation [14]. The algorithm continuously monitors the system for deviations from the normal behaviour patterns of milling operations. Monitoring of milling operations helps ensure safe operations and avoid possible damage to the machine tools. To monitor the machines tools, measurements of cutting parameters were collected, like: temperature cutting force, torque, vibrations, motor current etc. In these experiments, they found out that the proposed algorithm successfully detects the tool breakage, and can also detect noise in signals.

In [16], the authors worked on an immune-based intrusion (anomaly) detection technique for a computer network. In this work, they used the negative selection algorithm to detect *non-self* behaviour in a computer network, but also used a positive identification technique that is based on a nearest-neighbour classification technique. The experiments where performed on real intrusion data from the MIT Lincoln Lab, and the algorithms were able to recognize *non-self* behaviour in the network. After recognizing the anomaly, this was also preserved in long term memory for recognition of other behaviour patterns.

Similar work in [16, 23, 24, 54] shows AIS used for detection of computer viruses while [81] reveals AIS used for anomaly detection in refrigeration systems.

In [13], the authors developed a password authentication system that uses Negative Identification techniques as the first line of defence to identify legitimate users. Instead of using the original password data profile, a negative database is used to store Anti-Password detectors. Anti-P detectors are generated by pre-processing the original password data profile using a negative selection algorithm. This idea will be able to eliminate brute-force attacks on password database or servers. In their experiments, they found out that the Anti-P size grows almost linearly with the number of passwords and almost exponentially with coverage.

The work reviewed in this section is related to pattern recognition and anomaly detection, which are important in situation management for any system. The algorithms and techniques used in these experiments can further be used in situation management.

2.4. Summary

The literature review presented above only reports on the work done by applying each of the individual algorithms to a specific application field, and not applying the three algorithms on the same dataset to compare their performances. There has been literature comparing these algorithms with other existing methods. The remainder of this section presents a brief review of the literature and the work done in that direction.

[93] Proposed an Immune-inspired leaning technique for building a weather forecasting system. This was then compared to artificial neural network on an experiment that used weather data for April and May as training data for predicting the June Month (test data). Their results reveal that the AIS used for the experiment had a higher forecast accuracy of 81.0% relative to the 64.0% forecast accuracy of the artificial neural network.

In [46] Bayesian networks were applied to a problem of predicting sea breezes. In their experiments, Bayesian networks are compared to rule-based algorithms, revealing that the rule-based algorithms were outperformed by the Bayesian network in terms of predictive accuracy.

In [60] experiments were conducted to compare linear stochastic models: autoregressive integrated moving average (ARIMA) and seasonal autoregressive integrated moving average (SARIMA) with artificial neural networks. These models where applied to a system for forecasting drought using the standard precipitation index (SPI) series data from the Kansabati River Basin in West Bengal, India. Their experimental results revealed that the ANN is more suited for a longer prediction lead time of four months as it outperforms the ARIMA/SARIMA model. The ARIMA/SARIMA models produced good results with up to 2 months prediction lead time.

During our literature review, we did not come across a study comparing the three algorithms, AIS, Bayesian Network and ANN. In the light of the above, this dissertation sets out to investigate performance differences of three case studies based on the three algorithms, network monitoring, network intrusion detection and drought monitoring. The methods and application of the execution of this investigation are explained in the Research Design chapter which follows.

3. RESEARCH DESIGN

The aim of the thesis work is to evaluate performance of three machine learning algorithms and methods by comparing their performance on various datasets. This chapter provides a description of the steps and methods used to conduct research in this study, and the findings on the evaluation of the three machine learning algorithms on the following datasets:

1. Large-scale Wi-Fi Network Monitoring in Cape Town,
2. Large-scale Wi-Fi Network Monitoring with Network Intrusion, and
3. Rainfall, drought and Standard Precipitation Index data in Trompsburg (Free State, South Africa).

3.1. Research Method

There are various research methods available for one to choose from when conducting research. The nature and objectives of the research determine the research method one will use to come to a conclusion.

This research aimed to measure, analyse and compare the performance of three algorithms. This was done in the form of programming experiments on three datasets. The research methods used for this study therefore took the form of a combination of two well- established research methods, viz. case studies and experimental methodologies. These research methods are described below.

3.1.1. Case Studies

Researchers have used case study research over a number of years across a variety of disciplines. In [64] a review of a few definitions of a case study was made and the following definitions were used:

- An event, an entity, an individual or even a unit of analysis. It is an empirical inquiry that investigates a contemporary phenomenon within its real life context using multiple sources of evidence.
- Case studies are concerned with how and why things happen, allowing the investigation of contextual realities and the differences between what was planned and what actually occurred.

The authors in [64] further argues that a case study is not intended as a study of the entire system, but is rather intended to focus on a particular issue, feature or unit of analysis in order to understand and examine the system. It is useful where one needs to understand some particular problem or situation in depth, and where one can identify cases rich in information.

The case study research methodology has met with much criticism from the research community. Authors in [94] have identified that the case study research methodology has disadvantages in that it lacks scientific rigour and reliability. The research method has been blamed for "being sloppy and allowing equivocal evidence or biased views to influence the directions of the findings and conclusion." The most common criticism of the research method has been that it provides very little basis for scientific generalisation because of the small number of subjects used, and its dependency on a single case exploration, making it difficult to reach a generalising conclusion. Case studies tend to be microscopic because of their limited sampling cases [90]. However, [94] argues that establishing parameters and setting objectives of the research are more important than a big sample size in a case study.

There are however, advantages to using a case study methodology in research. Most of the data used in case study research are used within its own context, i.e. within the situation in which the activity takes place. In this way, the researcher can have a holistic view and understanding of the broader picture of the entire system. Case studies allow variations, therefore can allow for both quantitative and qualitative analysis of data. The use of multiple cases from multiple sources of evidence can lead to some form of replication, therefore allowing a comparative analysis of case studies and generalisation of findings and results in a case study research.

Looking at the definition and uses of a case study research methodology, it was found out that it fits the purposes and objectives of our research. The advantages and disadvantages of using the methodology were looked into and assessed.

3.1.2. Experimentation

In [83], experimentation is seen as a method for exploration and testing theory. They are used where theory and deductive analysis do not reach. They probe the influence of assumptions, eliminate alternative explanations of phenomena, and unearth new phenomena in need of explanations. In this way, they help with induction: deriving theories from observation.

In natural sciences, experimentation has a few drawbacks. In experimentation it becomes hard to identify and control variables that may have influence on results, and it is possible for one to have difficulties in coming up with a theoretical explanation of the results [5]. Other problems may include having the experiments based on unrealistic assumptions, or some researchers may be tempted to manipulate the data, or find it impossible to quantify a variable of interest [83].

Experimentations are mainly used in natural sciences, and according to [5], computer science is in fact, 'not a science' but a *synthetic*, an engineering discipline that is concerned with *making things*, be it computers, intangible products like algorithms or software systems. Therefore we can conclude that computer science is not a natural science.

Then again we ask the question: is experimentation suitable for computer science? In [83] the author states that if experimentation is ignored in the computer science discipline, progress is

hampered. He further highlights the benefits of experimentation in computer science based on the following points:

- Experimentation can help build a reliable base of knowledge and thus reduce uncertainty about which theories, methods and tools are adequate.
- Experimentation and observation can lead to new, useful and unexpected insights and open a whole new area of investigation. Experimentation can push into new unknown areas where engineering processes slowly, if at all.
- Experimentation can accelerate progress by quickly eliminating fruitless approaches, erroneous assumptions and fads. It also helps orient engineering and theory in a promising direction.

The advantages presented above have motivated us to follow an experimentation methodology in our performance study of the algorithms behind our three machine learning techniques.

3.2. Sample Data

The machine learning algorithms considered in our work are used to make performance analysis on current events and predictions on future events from current datasets, find hidden patterns in these datasets and predict future datasets. Training and testing of machine learning algorithms are done using sample data. However, this might lead to re-substitution error, where the same data used to train the machine learning algorithms are again used to test its performance [91], resulting in overly optimistic indicators of performance. It would be best to use a representative data of the total dataset for training and testing (illustrated in Figure 3.1). In this way, the data used for testing did not take part in the training of the machine learning algorithm.

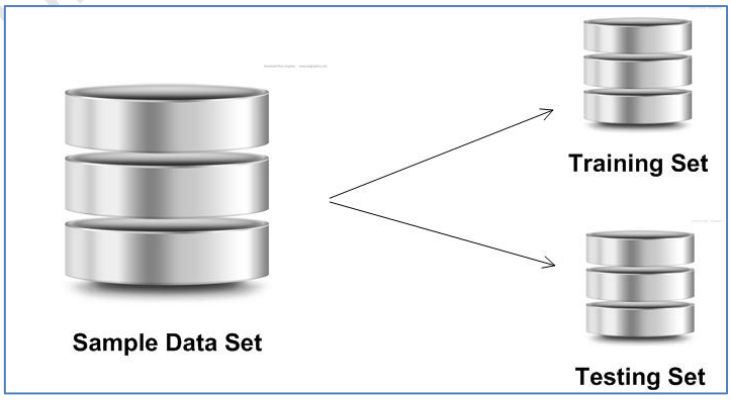

Figure 3.1: Sample Data split into Training and Testing set.

3.2.1. Challenges and Remedies of using sample data.

There are challenges facing researchers when using domain sample data for training and testing, the most important one consisting of avoiding overfitting and underfitting of data in the machine learning process.

3.2.1.1. Overfitting and Underfitting

Overfitting refers to the situation in which the algorithm used generates a model which perfectly fits the training data but has lost the capability of generalising or predicting for those representative instances that did not take part in the training of the algorithms. This means that any instance that is outside the training data will not be predicted correctly.

This occurs when the model is excessively complex, like having too many parameters relative to the number of instances in the training data. When the capacity is small relative to the number of instances in the training data, this is referred to as underfitting of data [74].

To avoid overfitting and underfitting of data, larger datasets and the use of machine learning techniques, such as the holdout procedure and the k-fold cross-validation, are required [72].

When working with large datasets on machine learning algorithms, one is faced with two critical challenges: space in memory and time. Space in memory is important since if the dataset is too large to be held in main memory for processing, it is difficult for the machine learning algorithms to process from such datasets. Time is another critical challenge since some machine learning algorithms do not scale linearly with the number of training instances, making it infeasible to train on large datasets.

When datasets are too large, instead of using the full dataset, one can apply subsampling on a small subset of the data for training. However, data is lost when using subsampling. Techniques such as the holdout procedure and k-fold cross-validation can also be used, making the loss of data negligible. The predictive performance of a learned model often flattens out long before all the training data is incorporated into it. This is a behaviour known as the law of diminishing returns.

The holdout procedure and the k-fold cross-validation are techniques that are generally accepted for subsampling in large datasets. Their description follows below.

3.2.1.2. Holdout Procedure

The holdout procedure reserves some data for training and the rest for testing. There may be instances where the data used for training or testing is not a representative of the full set, and the random-sampling is employed in such a way that all classes are represented in both training and testing. This is called stratification or stratified holdout procedure.

In this process a random sample is chosen for holdout, commonly two-thirds for training and the remaining for testing, and this process repeated and the error rates of the different tests are averaged to get the overall error rate [72, 91].

3.2.1.3. K-fold Cross-validation

Cross-validation is a statistical technique which is a variant of the holdout procedure, but in cross-validation you choose a fixed number of folds (K-folds) or partitions of the data. In this technique, training and testing are repeated until each partition is used for both training and testing. The error rates for all iterations are averaged to give the final error rate.

In practice, normally a 10-fold cross-validation is used where the data is randomly divided into ten partitions of the full dataset. Each partition takes part in the training and testing of the algorithm; this is done ten times. At each step the error rate is calculated and the average of all steps is used as final error rate [72, 91].

For the purpose of these experiments, the 10-fold cross-validation method is used.

3.3. Domain Knowledge

Machine learning algorithms perform best when the user knows and understands the domain in which the intended algorithms are to be used. It is therefore of utmost importance to get to understand the domain knowledge for the success of a machine learning algorithm.

Data can be classified in many forms, and in most statistics textbooks, data is defined as *nominal, ordinal, interval and ratio*. Nominal values are distinct symbols and they serve as labels for data. There is no notion for ordering or distance; in other words, there is no relation among the values. Ordinal values makes it possible to rank-order the categories; although there is ordering, there is no notion of distance and one can therefore compare the values [91].

Interval values can be ordered and measured in fixed and equal units. This allows the basis for measurement and comparison among the values. On the other hand, ratios are treated as real numbers, and are values for which measurement points inherently define a zero point [91].

To further understand the domain knowledge for which the use of machine learning algorithms is intended, one needs to know and understand the relations between the attributes of the data. Statistics books define three types of relationships among attributes, semantic, causal and functional.

In the sematic relationship of two attributes, if one attribute is included, the other should be there too. It is therefore known that the two attributes only make sense together. With a causal relationship, one attribute causes the other. So to predict the one attribute the other attribute must be included to make the prediction meaningful. In functional dependency relationships, the dependency of one attribute on another is that if the latter is used, there is no need to consider the former. If this relationship is not considered, the machine learning

algorithm may end up producing meaningless, tautological results that may obscure interesting patterns in the data.

It is important for one to consider domain knowledge of the data and context in which the machine learning algorithms are to be used; this reduces the possibility of difficult technical challenges.

3.4. Performance Evaluation of Machine Learning Algorithms

An important measure in machine learning is the measure of performance or evaluation criteria for machine learning algorithms. The performance evaluation criterion is very important to measure the quality of the function model for the chosen algorithm. There are several ways that can be used to measure predictive performance, but the error rate seems to be important. It would make sense to evaluate machine learning algorithms based on their success of prediction. The error rate measures the overall proportion of errors made in the total set of instances [72].

3.4.1. Measures of performance

In statistical tests, there are two types of errors, type 1 and type 2 [91]:

- Type I errors: these errors occur when a statistical test rejects a true null hypothesis. Also known as *false positive* (FP).
- Type II errors: these errors occur when a statistical test fails to reject a false null hypothesis. Also known as *false negative* (FN).

Other measures (which are desirable) exist, like the *true positive* (TP) which means rejecting the null hypothesis while it is not true; and the *true negative* (TN), which means not rejecting the null hypothesis while the alternative hypothesis is false.

Accuracy measures include the following:

3.4.1.1. Kappa Statistic

This is used to measure the success of a predictor, the agreement between predicted and observed categorisation of a dataset, while correcting for agreement that occurs by chance. [91]. It is calculated using the following formula:

$$K = \frac{\Pr(\text{observed}) - \Pr(chance\ agreement)}{1 - \Pr(chance\ agreement)}$$

3.4.1.2. True Positive Rate

This refers to the function of true positives out of the positive, measured by:

$$True\ Positive\ (TP)Rate = \frac{TP}{(TP + TN)}$$

3.4.1.3. False Positive Rate

This refers to the function of false positives out of the positives.

$$False\ Positive(FP)Rate = \frac{FP}{(TN + FP)}$$

3.4.1.4. Precision

Another important measure is precision, which measures how many instances classified as 'positive' class are indeed 'positive'. Examples in information retrieval theory: searching for relevant documents, Precision will represent the proportion of relevant documents in the results returned. It is measured as follows:

$$Precision = \frac{TP}{(TP + FP)}$$

3.4.1.5. Recall

This measures how well the classifier can recognise positive samples, for example in information retrieval theory, searching for relevant documents: Recall will represent the ratio of relevant documents found in the search result to the total of all relevant documents. It is measured using the formula below:

$$Recall = \frac{TP}{(TP + FN)}$$

3.4.1.6. F-Measure

There is a trade-off between Precision (P) and Recall (R) measures. When one tries to improve the first measure, there is often deterioration in the second measure. This is illustrated by the Figure 3.2 below:

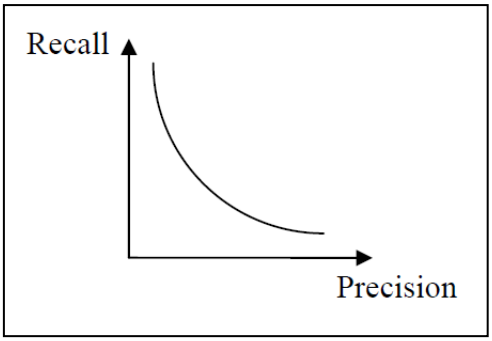

Figure 3.2: A typical Precision-Recall Graph

The problem here is described as a Multi-Criteria-Decision-Making (MCDM). The 'weighted sum model' is the common method to solve MCDM. The F-measure provides a harmonic mean for the MCDM problem, and it is defined by the formula:

$$F = \frac{2 \times P \times R}{P + R} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

3.4.1.7. ROC Area

Another measure used is the Receiving Operating Characteristics (ROC) curves; these indicate trade-offs between true positives rate and false positive rate. The ROC (graph) curve's x-axis represents false positive rate while the y-axis represents the true positive rate. The north-west corner of the curve is the ideal place.

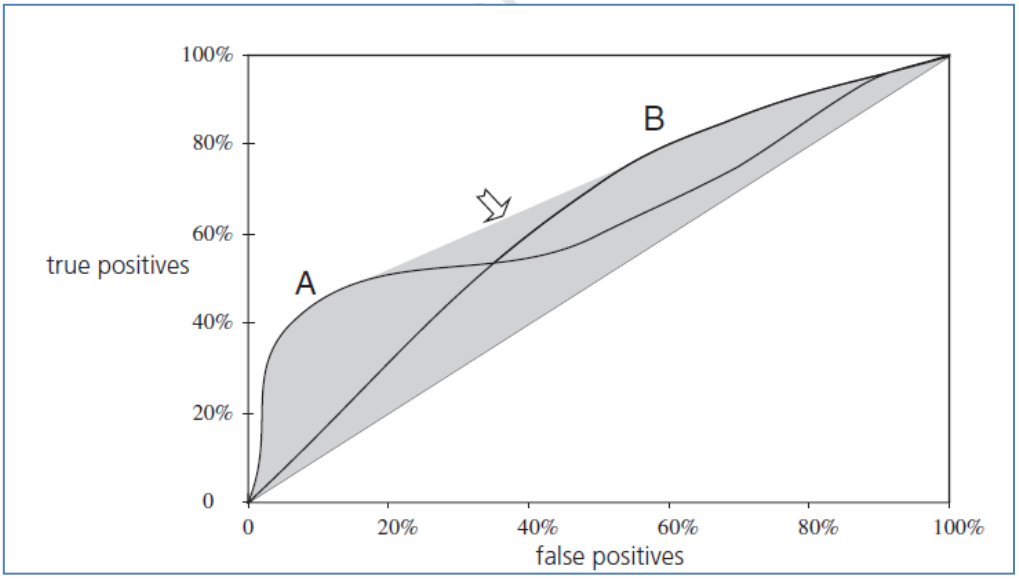

Figure 3.3: ROC curves for two machine learning algorithms (taken from [91])

The area under the ROC curve (AUC) is a measure of the probability of the classifier ranks a randomly chosen positive instance above a randomly chosen negative one. The larger the area under the curve the better the classifier.

3.4.1.8. Confusion Matrix

This is a multidimensional matrix with rows and columns for each class. It contains the number of elements that have been correctly or incorrectly classified for each class. In a confusion matrix, the main diagonal of the matrix indicates the number of observations that are correctly classified for each class. Therefore good results will correspond to large numbers down the main diagonal and small numbers, ideally zero, off-diagonal elements.

Table 3.1: A Typical Confusion Matrix

	Predicted Negative	Predicted Positive
Negative Example	A	B
Positive Example	C	D

3.4.2. Graphical Descriptive Techniques

To describe the algorithm performance measures mentioned above, graphical, numerical and tabular statistical methods that help to summarise data and to visually present useful performance information will be used. They help to quickly absorb information, observe trends and easily interpret data.

In this study, a combination of interval and nominal data is used. To better describe this kind of data, bar charts are used. Bar charts are usually simple and easy to read and understand; they enhance the reader's ability to grasp the substance of the data. They are excellent for data comparison, which is the main objective of this study, to compare performance measures for the three algorithms. For these reasons, the bar chart and tabular representative techniques are better suited for this study.

3.4.3. Evaluating Machine Learning Algorithms' Performance on the same Dataset

The present study aims to compare performance of the three machine learning algorithms on a sample problem to see which is better to use. In other words, the study plans to establish whether one scheme is better or worse than another machine learning algorithm on average, across all possible training and test dataset in that given domain.

To accomplish this, there exist statistical tests with confidence bounds that perform comparisons given two or more methods trained on the same dataset of the same size.

For the experiments conducted in this research, the repeated 10-fold cross-validation technique was used. Each cross-validation experiment yields a different, independent error estimate. Our interest is in whether the mean performance, on one machine learning

algorithm is significantly greater than or less than the mean of the other machine learning algorithm. The experiment was repeated 5 times for each test case. In total we had 200 (20*10-folds) iterations of the experiment. The mean algorithm performance measures of each experiment's iteration were recorded and used for comparison.

The notation, $(x_1, x_2 \ldots x_k)$ is used for one machine learning scheme, and $(y_1, y_2 \ldots y_k)$ for the other machine learning scheme. Their means will be represented by $\overline{x}$ and $\overline{y}$, respectively. To determine whether the mean, $\overline{x}$ is significantly greater or less than the mean, $\overline{y}$, we will consider the mean's difference, denoted as $\overline{d}$, $where\ \overline{d} = \overline{x} - \overline{y}$.

$$H_o : \overline{d} = \overline{x} - \overline{y} = 0$$

$$H_1 : \overline{d} = \overline{x} - \overline{y} \neq 0$$

The null hypothesis states that there is no difference in the two means, while the alternative hypothesis states that there is a difference in the means. The two-sided confidence estimate of the mean difference will be used [91].

This paired student t-test is used for this purpose and it is represented by the formula:

$$t = \frac{\overline{d}}{\sqrt{{\delta_d}^2/k}}$$

Where ${\delta_d}^2$ represents the variance difference of $\overline{x}$ and $\overline{y}$ samples.

The variance is estimated and therefore has a student's distribution with *k-1* degrees of freedom. Using a given confidence bounds, the null hypothesis can be rejected or the alternative hypothesis cannot be rejected [18, 72].

For the purposes of this study, the t-statistical test is conducted at the 5% significance level and its p-value is then presented. The p-value of a test is the probability of observing a test statistic at least as extreme as the one computed given that the null hypothesis is true.

3.5. Software Program and Packages

As part of the research, software programmes were written, and some already existing software programs were used for the evaluation of the three machine learning algorithms and how each of the machine learning algorithms perform on each of the three datasets.

The programs written for this study were written in the Java language. Java is a simple language designed to be easy to write, compile and debug. It is an object-oriented language centred on the creation, manipulation and collaboration of objects. In this way, Java allows the creation of modular programs and reusable code, and it is platform-independent.

Weka 3.0 [25], an open-source Java-based machine learning machine software program was used in the experiments. Weka stands for Waikato Environment for Knowledge Analysis; it was developed at the University of Waikato in New Zealand. It is a collection of state-of-the-art machine learning and data pre-processing tools. It is designed for experimental data mining. It allows one to use methods on new datasets, and it allows statistical evaluation of various machine learning schemes on various datasets.

Microsoft Excel's Analysis ToolPak is a set of analysis tools provided in the Microsoft Excel installation. They are used when developing complex statistical or engineering analyses. When provided with data and parameters for each analysis, the tools use appropriate statistical or engineering macro functions and then display results in output tables [59].

3.6. Algorithms Parameter Settings in Weka

The Weka software toolkit has various machine learning algorithms implemented for data mining. In the Weka system one will need to fine tune the parameters of each algorithm and the section below will give a detailed description of the Algorithms' parameters used in the experiments [78, 88].

3.6.1. Multilayer Perceptron

For Artificial Neural Networks, the Multilayer Perceptron classifier with backpropagation was used for data mining. This algorithm has parameters that can be fine-tuned. They are described below.

- **GUI:** Brings up a Graphical User Interface (GUI) interface. This will allow the pausing and altering of the neural network during training. It provides interface for users to connect a new node, select a node, disconnect a node, deselecting a node, etc. This variable takes up a Boolean value: True or False.
  For the purpose of these experiments, the value was set to false.
- **Auto-Build**: This parameter automatically adds and connects up hidden layers in the network. It takes up a Boolean value: True or False.
  For the purpose of these experiments, the value was set to true.
- **Debug**: If set to true, classifier may output additional info to the console. For the purpose of these experiments, the value was set to False.
- **Decay**: This will cause the learning rate to decrease. This will divide the starting learning rate by the epoch number, to determine the current learning rate. This may help to stop the network from diverging from the target output, as well as improve general performance. The decaying learning rate will not be shown in the GUI, only the original learning rate. If the learning rate is changed in the GUI, this is treated as the starting learning rate.
  For the purpose of these experiments, the value was set to False.

- **Hidden-Layers**: The hidden layer of the neural networks is defined by this option. This parameter takes up positive whole numbers, separated by comma, for the number of nodes in each hidden layer. It is also possible to use one of the wildcard values:
    - **'a'** = the average of the number of attributes plus the number of classes,
    - **'i'** = number of attributes,
    - **'o'** = number of classes,
    - **'t'** = the number of attributes plus the number of classes.

  For the purpose of these experiments, the value parameter was set to wildcard **'a',** which is the average number of attributes plus the number of classes.
- **Learning-Rate**: This determines the amount by which the weights in the network are updated. The parameter takes up number values between 0 and 1.
  For the purposes of these experiments, the value was set to 0.3.
- **Momentum**: This option defines the momentum that is applied to the weights in the network when it is updated. The parameter takes up a number values between 0 and 1.
  For the purposes of these experiments, the value was set to 0.3.
- **Nominal To Binary Filter**: This parameter is used for preprocessing the instances with the filter. If there are nominal attributes in the data, the performance of the network may improve by preprocessing the instances with the filter. The parameter takes up a Boolean value: True of False.
  For the purpose of these experiments, the value was set to true.
- **Normalize Attributes**: With this parameter the attributes will be normalized. The performance of the network may improve when the attributes are normalized. This is not reliant on the class being numeric. This will also normalize nominal attributes as well (after they have been run through the nominal to binary filter if that is in use) so that the nominal values are between -1 and l.
  For the purpose of these experiments, the value was set to true.
- **Normalize Numeric Class**: The class will be normalized if it is numeric by this option. The performance of the network may improve when the class is normalized. The class will be normalized between -1 and 1. This is only internal; the output will be scaled back to the original range.
  For the purpose of these experiments, the value was set to true.
- **Reset**: This parameter resets the network with a lower learning rate when the network diverges from the target output. When this parameter is set to false and the network diverges from the answer, the network will fail the training process and return an error message.
  For the purpose of these experiments, the reset value was set to true.
- **Seed**: This parameter defines the seed that is used to initialize random generator. Random numbers are used to set the initial weights of the connections between nodes and for randomizing the training data.
  For the purpose of these experiments, the initial seed value was set to 0, zero.
- **Training Time**: This defines the number of epochs that the network will be trained. If the validation set is non-zero then it can terminate the network early.
  For the purpose of these experiments the training time is set to 500.

- **Validation Set Size**: The percentage of the training set that will be used for the validation set is determined by this parameter. The training of the network will stop when the times that the error on the validation set gets worse in a row reaches the value of Validation-Threshold. When this parameter has the value 0, the network will be trained using the value of Training Time.
  For the purpose of these experiments the validation set size is set to the value 0, zero.
- **Validation-Threshold**: This parameter is used to terminate validation testing. The value here dictates how many times in a row the validation set error can get worse before training is terminated.
  For the purpose of these experiments the validation threshold is set to the value 20.

3.6.2. Bayesian Belief Networks

For Bayesian Belief Networks, the popular NaiveBayes algorithm was used for experiments. The Weka system has the NaiveBayes implementation and does not have many parameters to configure. The default Boolean parameter settings were used for the purposes of the experiments.

3.6.3. Artificial Immune Systems

For Artificial Immune Systems, the Artificial Immune Recognition System Algorithm (AIRS2) was used. The algorithm had the following parameters:

- **Affinity Threshold Scalar**: **Affinity** is a measure of "closeness" or similarity between two antibodies or antigens. In the current implementation of AIRS, this value is guaranteed to be between 0 and 1 inclusively and is calculated simply as the Euclidean distance of the two objects' feature vectors. Thus, small affinity values indicate strong affinity.
  The A**ffinity Threshold** is the average affinity value among all of the antigens in the training set or among a selected subset of these training antigens. The **Affinity Threshold Scalar** is a value between 0 and 1 that, when multiplied by the affinity threshold, provides a cut-off value for memory cell replacement in the AIRS training routine.
  The measurement provides a means of adjusting the automatic threshold by making it softer (less than the mean) or harder (more than the mean). The effect of softening the threshold causes less replacement of best matching memory cells by candidate memory cells, and the reverse is true when the threshold is hardened. Common values for this user parameter are in the range [0.1, 0.3], which is a significant softening of the mean.
  The effect of having the scalar too close to the mean is that too many replacements occur, thus squashing the threshold down by to a factor of 10% or 20% of the mean causes less replacements, and thus a larger, more effective classification memory pool.

For the purpose of these experiments the affinity threshold scalar is set to the value 0.2.

- **Clonal Rate**: an integer value used to determine the number of mutated clones a given cell is allowed to attempt to produce. In the current implementation, a selected cell is allowed to produce up to (clonal rate * stimulation value) mutated clones after responding to a given antigen. This product is also used in assigning resources to a cell. Therefore, the clonal rate serves a dual-role as resource allocation factor and clonal mutation factor for the cell population. A typical value for the clonal rate is approximately 10.
  For the purpose of these experiments the clonal rate is set to the value 10.0.
- **Hyper-Mutation Rate**: an integer value used to determine the number of mutated clones a given memory cell is allowed to inject into the cell population. In the current implementation, the selected memory cell injects at least (hyper-mutation rate * clonal rate * stimulation value) mutated clones into the cell population at the time of antigen introduction.
  For the purpose of these experiments the hyper-mutation rate is set to the value 2.0.
- **K-Nearest Neighbours**: This parameter is only used during the read-only classification stage of the algorithm. As has been mentioned, it determines the number of best match memory cell s used to vote by majority on the classification of unseen antigens (data vectors). When a tie occurs in the majority vote, the class index with the lowest number is always selected, making classification deterministic (as opposed to probabilistic or stochastic tie breaking strategies). Common values for the kNN are in the range of [1, 7].
  For the purpose of these experiments the k-Nearest Neighbour is set to the value 3.
- **Stimulation Value**: This parameter controls the amount of refinement performed on ARBs for an antigen, and thus how closely the ARBs will be to the antigen in question. Stimulation values are commonly high, around 0.9. This means that the mean stimulation value must be quite high, that is the vast majority of the ARBs in the pool must be similar to the antigen. The range for the stimulation value must obviously be in the range of [0, 1], given the mean also will have the same range.
  For the purpose of these experiments the stimulation value is set to the value 0.9.

4. REDBUTTON WI-FI NETWORK MONITORING CASE STUDY

4.1. Introduction

Wireless Fidelity (Wi-Fi) is a wireless networking technology that uses radio waves to provide high-speed wireless internet connections. It is based on the family of IEEE 802.11 standards and builds upon a fast, easy and inexpensive networking approach [87] that uses a client-server model where the access point (AP), also called a hotspot, plays the role of server while its client devices range from laptop computers, IPADs, tablets and Personal Digital Assistants (PDAs) to smart phones. The Wi-Fi APs broadcast signals to Wi-Fi-capable client devices that detect and receive broadcast messages from hotspots within their AP's range, and thus connect to the Internet.

It has been reported that in 2003, 15 million Wi-Fi adapters for computers and 4.4 million access points were sold worldwide. The chart in Figure 4.1 shows that in 2003, there were 20 000 hotspots in private locations while there were above 60 000 hotspots in public locations with a predicted growth of just above 60 000 hotpots at private locations while hotspots in public locations experienced a growth of just above 140 000 hotspots in 2005. This reveals that overall, private hotspots are outnumbered by public hotspots which are growing at a fast rate. In their report, Gartner reveals that in 2004 the number of hotpot users tripled and totalled 30 million worldwide and more than 50 percent of notebooks had Wi-Fi capabilities [27, 82]. These predictions by Gartner Inc. [28] became a reality and today, almost every notebook is equipped with a Wi-Fi card, and smart mobile phones are equipped with a Wi-Fi access card. Initially deployed in private dwellings such as homes, schools, universities and offices, Wi-Fi has shifted to public hotspots. Firms like (USA's) T-Mobile, AT&T wireless and (in this case study) RedButton Mobile CC have rushed to install public hotpots in business locations, airports, hotels, shops and restaurants.

In the first quarter of 2011, 428 million mobile devices were sold worldwide, indicating a 19 percent year-on-year increase in mobile device sales. In the first quarter of 2011 smartphone sales were 23.6 percent and they are predicted to overtake the market and drive smartphones to mass adoption. Smartphone's sales had an 85 percent year-on-year increase. This may increase the need for more public Wi-Fi hotspots, as predicted by Gartner Inc. [27]. This presents a great business opportunity for the Wi-Fi hotspot internet service providers.

The increased demand for mobile access will drive enterprises (clients) to reduce the number of wired networks. This will lead to deployment of large network of Wi-Fi hotspots. The Wi-Fi hotspots internet service providers will enter into a service level agreement (SLA) with the clients. Performance monitoring is an important task upon which large Wi-Fi networks depends. This is to ensure that Wi-Fi hotspots internet service providers meet the SLA entered with the clients.

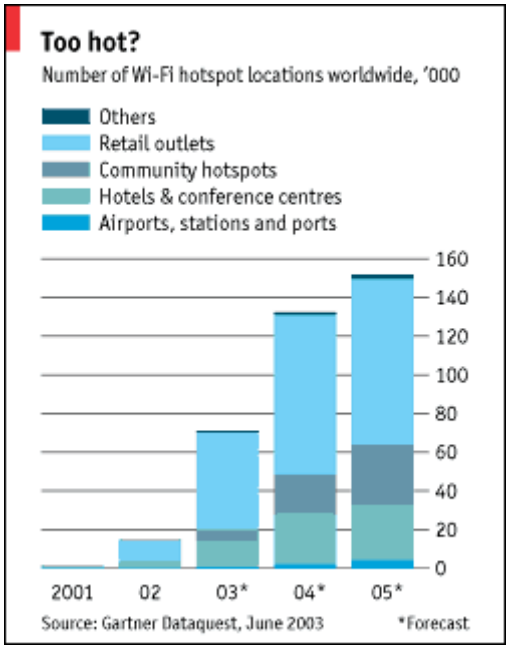

Figure 4.1: Graph showing worldwide public hotspots per location

The popularity of Wi-Fi technology has led to a large scale deployment of thousands of hotspots networks generating huge amounts of monitoring data which require efficient data handling methods to analyse and recognise anomalous hidden patterns and implement fault tolerance. As traditionally implemented, performance monitoring is based on a reactive network approach where the operating system software only warns the network administrators when a problem occurs. This approach leads to both the halting of important network processes and the hampering of critical business processes of the organization. Pre-emptive network monitoring provides the potential to prevent the occurrence of faults by analysing the status of the network components to create a fail-safe network status or allow a smooth migration from a faulty to fail-safe network status. While statistical analysis has been deployed in many cases to address the performance monitoring issue, soft computing methods are emerging as powerful tools used in anomaly detection and security monitoring systems.

4.2. Background and Related Work

There has been work done in the field of AIS, which has mostly focused on intrusion detection, detection of computer viruses and network security [54], [23], [24], and [16]. While [81, 81] addresses anomaly detection for a refrigerator system, the works in [23] and [24] describe AIS mechanisms to differentiate between self and non-self performance with application to a dataset of executable files infected by computer viruses.

In [13], a security authentication system inspired by the immune system using the negative selection algorithm is introduced. This system generates a set of anti-passwords (a negative image of the password set) used as a first line of authentication kept separate from the positive authentication system (secured). Using datasets with approximately 2500 most

common passwords, the authors generated a set of anti-passwords that defined the non-self of the password authentication system with the objective of providing a proof-of-concept for negative authentication systems using AIS to reduce the effect of brute force attacks on authentication systems by hackers. The tool breakage detection tool developed in [14] aims to detect tooth breakage in different environments. An implemented negative selection technique is successfully used to detect the tooth breakage from dynamic variations of the cutting force signal.

There has also been work done in the field of ANN relating to anomaly and intrusive behaviour detection. Experiments in [8] were conducted to find out if neural networks were able to recognise activities that represent Denial-of-Service attacks. In this work he evaluated the ability of neural networks to recognise new patterns through generalisation. The experiments revealed that neural networks achieved a 3.24% error rate, which was significantly better than the 15% achieved by the commercial intrusion detection systems. In these experiments, the DARPA-Lincoln and the KDD99 datasets [92] were used for benchmarking and comparison.

Feed-forward neural network were used in [79] to distinguish between normal and abnormal behaviour in UNIX computer systems. In the experiments the authors used information such as command sets, CPU usage, and login host address to investigate the effectiveness of neural network at modelling user behavioural patterns. Authors in [95] used self-organising maps for detecting intrusive and anomalous behaviour in Unix servers. These maps are also known as the Kohonen maps, which are single-layer feed-forward neural networks. In their experiments the authors used a *tcpdump* of packet-capture program that is standard on Unix to monitor requests and replies to their Domain Name Server (DNS).

In [22], the authors developed a system, BATmobile for Intelligent Vehicles and Highway Systems (IVHS), which aims to substantially reduce congestion and accidents on the roads. They used Bayesian Networks for decision making given different conditions that were observed.

The authors of [45]used Bayesian Belief Networks to develop the System for Automated Customer Support Operation (SASCO). This system was developed in partnership with Hewlett-Packard for trouble-shooting printing systems. Printers are complex systems with several components and trouble shooting is not a trivial task. The SASCO system automates this process.

In [2] the authors developed a Bayesian Belief Network, **HailFinder**, which combines meteorological data and BBN models to forecast severe weather in North-Eastern Colorado. This was one of the first forecasting systems developed using BBN techniques.

Building upon the success of the AIS, NaiveBayes and ANN we would like to implement the three machine learning algorithms on a large network of Wi-Fi hotspots and answer the following research questions:

- Which machine learning algorithm performs better?

- How do they perform under different test cases and network thresholds?
- Can these methods detect anomalous and aberrant behaviour?
- What kind of information can be mined using these machine learning algorithms?

4.3. The Experimental Model

This experiment in this case study was conducted as part of project for developing a network-monitoring tool built to achieve pre-emptive monitoring of a large-scale live ISP network operating in the city of Cape Town in South Africa. This section of the chapter will give a description of the case study experiment conducted. This includes the experimental network and the data collection method and encoding of the data entries used in the experiment.

4.3.1. The Wi-Fi Network

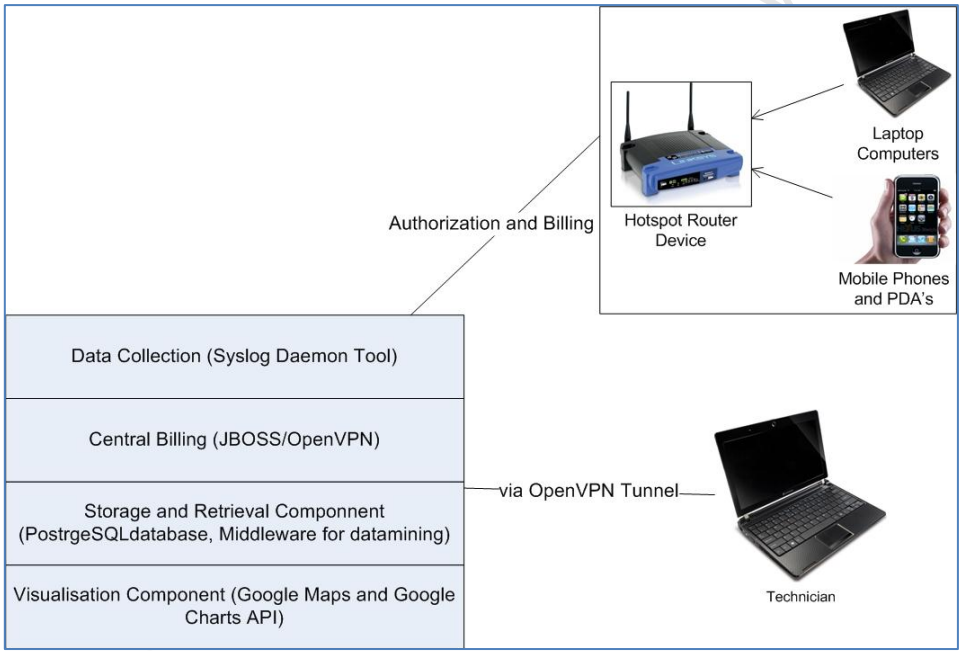

Figure 4.2: The Wi-Fi Network Monitoring Tool.

As depicted by Figure 4.2, a Wi-Fi network monitoring tool was developed around WRT54GL routers with three components: a storage component, a visualisation component and a data collection component. The 'Storage component' provides storage for the enormous amount of data that is harnessed during network monitoring. A postgreSQL database was used for this component. The "Visualisation Component" provides visual display of the performance of the network. Google Maps was used as a visual tool as the network monitored covered a large metropolitan city of Cape Town. Google Chart Tools were used to visualise performance statistics and individual router performance. Of most importance was the 'Data Collection component' which provided monitoring and data-gathering capabilities for the monitoring tool. The details of the 'Data Collection Component' follow below.

4.3.2. Data Collection Methods

For the network monitoring tool, two data collection methods were identified for use in the network, The Simple Network Management Protocol (SNMP) and Syslog protocols. The section that follows briefly describes the data collection methods, and looks at the advantages and disadvantages of using each method. This investigation will include cost or data overheads for each data collection method. As in performance monitoring, bandwidth is a scarce and very expensive resource, a data collection method that is not bandwidth intensive is preferred to a bandwidth-intensive method. Such method will reveal less overhead when collecting data

4.3.2.1. Simple Network Management Protocol

SNMP (RFC1157) is a protocol that is defined by the Internet Engineering Task Force. It is used in network management system for monitoring devices that are connected to the network. It is used to pull and receive SNMP information from SNMP-enabled devices that are attached to the network, providing collection services useful to any administrator. It uses a centralised client-server model, as illustrated in Figure 4.3; it allows management systems (software collecting the SNMP information) to poll agents running on the network devices (switches, routers, gateways, etc.) for specific pieces of information. This information may include bandwidth, uptime, temperature on device, and many other metrics that can be used [77].

All SNMP-enabled devices contain a specific text-file called the Management Information Base (MIB). It is a collection of hierarchically organised information that defines what specific data can be collected from that particular device. SNMP will access information on a particular device that its MIB describes. MIB compilers convert these text-based MIB modules into a format usable by SNMP management stations. With this information, the SNMP management station queries the device using different commands to obtain device-specific information. SNMP is non-proprietary and commonly used by network administrators. It is a standard, one-management system that can communicate with devices from multiple vendors [77].

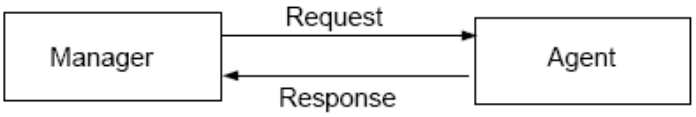

Figure 4.3: Centralized Client-Server Model

4.3.2.1.1. The SNMP Message

SNMP is simple because it uses five basic messages to communicate between the manager and the agent [77]:

- **GetRequest** and **GetNextRequest**: This allows the manager to send a request for information for certain variables; thus, it reads information from the agent;
- **SetRequest**: This allows the manager to request a change in a specific variable of the agent; this information is then updated to the agent's MIB;
- **GetResponse**: This is sent by the agent to the manager in response to a **GetRequest**, **GetNextRequest** or a **SetRequest** message. This message contains the requested values, or it confirms that changes were made, or it may contain any error indication of previous requests; and
- **Trap**: This type of message allows the agent to asynchronously notify the manager of any significant event or changes.

The SNMP message has a variable binding part (variable size of message); this allows the manager to group a number of operations (get, set, trap) into a single message (sending a single message requesting all values at a single agent).

4.3.2.1.2. SNMP Security

SNMP has the following security threats when used:

- Masquerade **–** An unauthorised entity may attempt to perform management operations by assuming the identity of an authorised entity.
- Modification of information **–** An entity may alter an in-transit message generated by an authorized entity
- Message Sequence **–** Messages may not be received in sequence, they may be delayed or duplicated.
- Denial-of-Service – An attacker may prevent exchanges between a manager and an agent

Mitigation Strategies:

SNMP has the following security services [77]:

- Data Integrity – ensures that the messages are received as sent, with no duplication, insertion, modification, sequencing or replays.
- Data-origin authentication **–** provides for the confirmation of a messages source
- Data confidentiality – this assures that the information is not made available or disclosed to unauthorised individuals, entities or processes.

4.3.2.1.3. SNMP Overheads

The SNMP version only consumes 1 byte. The Community String is used as a relationship between the manager and the agent. It defines authentication, access control and proxy

characteristics. Each community is given a name that is unique, within the agent, and this is included in each message. The Community String has a length of 7 bytes [77].

The PDU type defines the type of messages that is sent: this is 1 byte [0-4] [77].

- 0-GetRequest;
- 1-GetNextRequest;
- 2-GetResponse;
- 3-SetResponse;
- 4- Trap.

The SNMP PDU starts with the Request ID, which is a 4-byte integer value that uniquely identifies each query sent to the device, and the response received from the device should contain the matching Request-ID.

Error-Status and the Error Index are both 1-byte integers. They contain a zero for a GetRequest PDU, and for others PDU's it will indicate whether an error has occurred in the SNMP transaction, and what kind of error has occurred.

Variable-Binding is a sequence of two specific fields. The first field is the Object ID (provided by the MIB), which addresses a specific parameter; the second field contains the value of that specific parameter. The Object ID is a list of numbers separated by periods, for example '1.3.1.6.1.4.1.2680.1.2.7.3.2.0' and this is calculated in a special way such that it fits in 13 bytes. When sending a request (GetRequest and a GetNextRequest) to the SNMP agent, the value of the specific parameter is set to null; the value will change with a GetResponse and SetResponse.

The value can be an integer and a string or can take any form; therefore it has a variable size. Now let us look at how messages may be presented in our project. In one message we can request more than one parameter, so we are going to use the bulk-sending functionality of SNMP. Every hour, the SNMP Manager will request the following data from the SNMP agent: Uptime, Load Average, Radio Noise and Wireless Scan.

Therefore there will be a sequence of four parameter and values in the SNMP message. Therefore message size for a GetRequest: SNMP version. + Community String + PDU Type + Request-ID + Error-Status + Error-Index + Variable Binding 1 byte + 7 bytes + 1 byte + 1 byte + 1 byte + 1 byte + 1bytes + 4*(13 + 0) = 65 bytes in one direction The GetResponse message from the agent to the manager: The agent will respond with the values of the requested parameters.

Each parameter value may be in different types (integer, string etc.); the integer is 10 bytes (maximum number) while a string can be less than 255bytes (maximum length of a string). The values we are trying to pull from the SNMP agents are integer numbers; therefore they should not exceed 10 bytes per value. Therefore the message size should be: 1 byte + 7 bytes + 1 byte + 1 byte + 1 byte + 1 byte + 1 byte + 4*(13 + 10) = 105 bytes in one direction.

These messages are wrapped in a UDP packet for transportation, and a UDP packet is made of four fields: the Source Port (4 bytes), Destination Port (4 bytes), Length (8 bytes), the Checksum (2 bytes) and the data part (total is 18 bytes).

Therefore total one-way length of a message is 65 + 18 bytes = 83 bytes, while the length of a response is 105 + 18 bytes = 123bytes. The total bandwidth used to pull data from an SNMP agent is 83 bytes + 123 bytes = 206 bytes.

We assume a 31-day month, and these measurements will be pulled every hour, therefore: 31 days * 206 bytes message/hour * 24 hours/day = 153264 bytes (~149.7KB = 0.14MB) per month per month.

4.3.2.2. Syslog Protocol

Syslog (RFC5424) is also a client-server protocol that provides a framework under which machines (agents) can send event notification messages across an IP networks to event message collectors – also known as Syslog Servers or Syslog Daemons. Each message is a single line of text with an associated facility and severity. The facility can be thought of as a category that depends upon the program from which the message originates. Severities are hierarchical and range from the most important down to the least significant. The facility and severity of the Syslog protocol allows one to define how particular severities and facilities of messages are logged.

Initially Syslog messages are stored locally; and these messages will be automatically routed to a central location. These messages are received by the logging host; the logging host has significant disk storage for incoming messages (stored in a database) [29, 53].

4.3.2.2.1. The Syslog Message

The Syslog message body has three parts [69]:

- PRI – this represents the priority and the severity and facility of the message (identifies the source of the message).
- Header – this part of the message consists of timestamps followed by an indication of the hostname IP address of the device.
- Message (MSG) – contains additional information of the process that generated the message and the text of the message.

According to the [69], the maximum size of a Syslog message is not more than 1024 bytes (1KB).

4.3.2.2.2. Syslog Security

The Syslog message can be encrypted through Secure Socket Layer (SSL) or Transport Layer Security (TLS) to provide security and data integrity.

- Message length – In Syslog, the message length must not exceed 1024 bytes; if the message is greater than 1024 bytes, the receiver may malfunction upon receipt of the message. The received message may be logged completely, partially, or just discarded altogether [29, 53].
- Message Authenticity – The Syslog delivery mechanism does not strongly associate the message with the message sender. The receiver of that packet will not be able to ascertain that the message was indeed sent from the reported sender, or if the packet was sent from another device [29, 53]. This drawback of Syslog may lead to Denial-of-Service (DoS) attacks.
- Message Forgery – Malicious exploits of this behaviour has also been noted. An attacker may transmit Syslog messages (either from the machine from which the messages are purportedly sent or from any other machine) to a collector [29, 53]. This can also lead to DoS attacks.
- Message sequencing – The Syslog process and protocol do not ensure ordered delivery.

4.3.2.2.3. Syslog Overheads

We now look at the overhead and the bandwidth consumption of the Syslog method. An example of a Syslog packet:

"stat: version=v1.0beta2 lanip=172.25.0.1 noise=-9405/4 uptload=1:52/0.01 usage=2532952/3876804"

This packet contains the following data: firmware version, IP address of LAN interface, noise (average over an hour), and uptime/15 minutes load average, and usage (Receive/Transmit of a WAN interface). The total length of this UDP packet is 142 bytes. Therefore the total bandwidth used in a month (also assuming a 31-day month) is: 31 days * 142 bytes message/hour * 24 hours/day = 105648 bytes (~103.2KB = 0.10MB) per month.

4.3.2.3. Denial-of-Service

This is a big issue when coming to both methods of data collection. For these collection methods, there is no formal protocol or way of avoiding denial-of-service. This is justified on two grounds: first, denial-of-service is in many cases indistinguishable from the type of network failures with which any viable network-management application must cope as a matter of course; and second, a denial-of-service attack is likely to disrupt all types of exchanges and is a matter for an overall security facility, not one embedded in a network-management protocol (56). Denial-of-service can be defeated by constraining an attacker's consumption of resources. There are techniques available that constrains denial-of-service

attackers to a small percentage of system resources and that would slow such attacks sufficiently so that they can be detected.

- We can implement a router filter that lessens the exposure to certain denial-of-service attacks.
- We can also implement redundant and fault-tolerant network configurations for routers.

4.3.2.4. Conclusion

From the above investigations, Syslog seems to be bandwidth-efficient as it consumes only 0.10Mb in a month for collecting the data from routers, while SNMP uses 0.14Mb per month of bandwidth. Security issues like denial-of-service apply to both SNMP and Syslog. Methods are available to reduce the impact of denial-of-service for both collection methods. In this project Syslog data collection method seems to be more efficient in bandwidth usage.

The data gathering component was implemented using the Syslog protocol, a syslog daemon program was installed on each of the 615 Cisco WRT54GL router devices in more than 400 hotspots in the Cape Town network. The program was left to run from 2009-07-03 12h00 to 2009-09-02 04h00 collecting monitoring data at every hour's interval, thus leading to up to 356537 items in the experiment dataset.

4.3.3. Performance Metrics

We conducted a set of experiments based on three performance metrics which are usually used in performance evaluation of Wi-Fi networks. These include:

4.3.3.1. Uptime and Downtime

This metric measures the time a device has been up and running. It reveals the availability, stability and reliability of the communication device when used in the network. The metric can help in measuring the downtime. The time the device was not available on the network. It is also a measure of the reliability and the stability of the network device; measuring the time the device can operate without needing attention. The device can experience unplanned power failures and when the device is down, it makes it impossible for the client to connect to the hotspot. The device can also go down due to scheduled maintenance plans or system reboots. In the SLA that the client and service provider may agree on, for example: 99.999% availability, reliability and stability of the Wi-Fi network's devices. The administrator can use the performance metric to measure if the agreed level of service is met, and if it is not met, how can they remedy the problem such that the SLA is upheld [26].

4.3.3.2. Load Average

Measures the 'congestion rate' for the device based on the number of users connected to the device. The routers or hotspots are placed at strategic business places like coffee shops, airports and hotels. There may be hotspots in positions where there are a large number of clients accessing the hotpot. If the hotspot becomes highly congested, some clients may not be able to access the hotspot (which is not good for business), simply because the router cannot handle the load on that particular hotpot. The historic data for this variable will allow the network operator to determine if the router at a particular hotspot is able to handle the load [26].

4.3.3.3. Radio Noise and Channel

Wi-Fi uses the 2.4 GHz spectrum band which is shared with other devices like cell phones, GPS, RFID tags and Bluetooth devices. Note that the proliferation of devices using the free 2.4 GHz ISM band leads to more congested and noisy Wi-Fi devices.

In a certain area, one can have a number of devices that use the shared frequency spectrum, and these devices may have overlapping channels, causing radio noise among them. This can be a problem in high-density areas, like airports and office buildings where there are many Wi-Fi access points and many users with devices that use the shared spectrum.

This level of noise can be very risky for the internet service provider; if the level of noise is very high, the client will not be able to connect to the hotspot's access point, and if they do connect, the quality of service will be very poor. Environmental factors can also cause a network device to experience high levels of noise; for example, road works outside an office may affect the noise level of a router of an organisation. The numerical data and graphs that are presented to the network administrator can help them recognise a trend in the level of noise for a router; they can then take appropriate actions to remedy the problem [26].

The section that follows will describe the experiment's planning and mapping

4.3.4. Experimental Design

The experiment plan and mapping consisted of four stages and they are shown by Figure 4.4 and explained below:

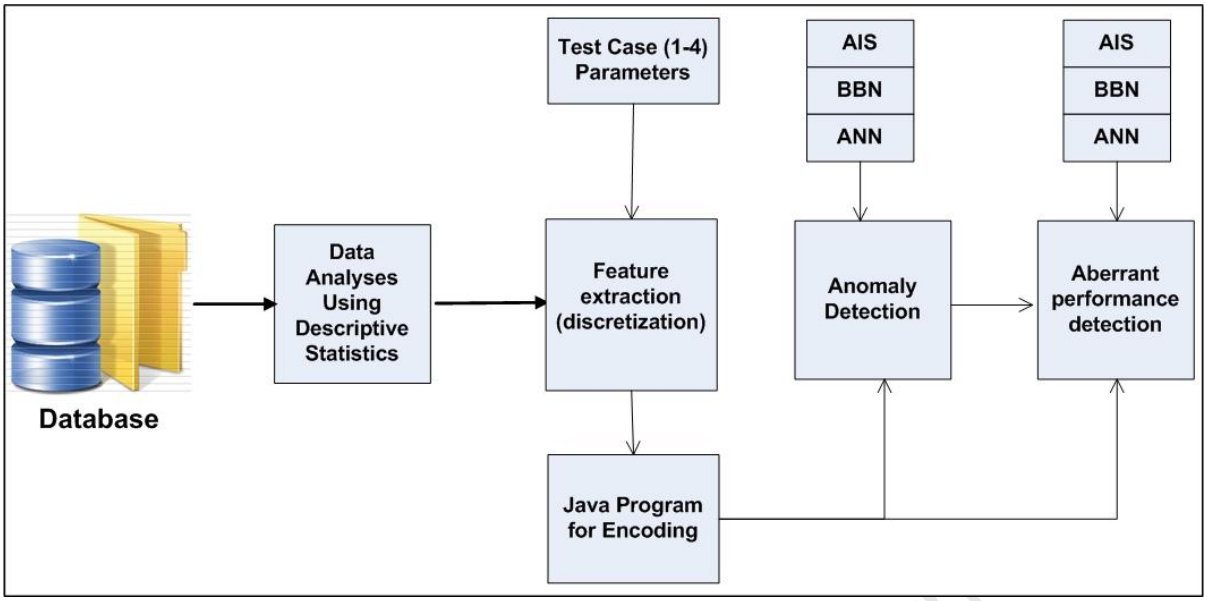

Figure 4.4: Experimental Plan

1. Data analysis. In this stage, descriptive statistics from the dataset were used to reveal the network's performance level in terms of time series characteristics and descriptive summary statistics. This allowed a summary of the observations by considering their central tendency and statistical dispersion.
2. Feature extraction. At this stage, the readings/observations received were discretised or reduced for further processing.
    a. A Java program was written specifically for feature extraction. The Java program takes raw data from the database, together with the information revealed in the 'Data Analysis Stage' to perform feature extraction.
    b. This program was written also for preparing and encoding database entries into data entries that will be used by the algorithms for anomaly detection and aberrant behaviour detection.
    c. For encoding the database entries, the Java program used the test case parameters supplied. The method of encoding is explained in section 3.5 and the test cases parameter are described in section 3.6.
3. Anomaly Detection. This stage is aimed at discovering those readings/observations associated to unacceptable device performance. The data produced at this stage were then used by machine learning algorithms AIS, BBN and ANN to perform anomaly detection.
4. Aberrant Performance Detection. This stage uses the observations obtained from the 'Data Analysis stage' to detect aberrant behaviour. An observation will fall into the aberrant behaviour category when its observed values fall outside the statistical confidence band, an outlier. A total deviation of the observation will depend on a defined Delta (δ) parameter. The data produced at this stage were fed into machine learning algorithms for further processing and aberrant performance detection AIS, BBN and ANN to perform aberrant behaviour detection.

4.3.5. Encoding of Performance Metrics

Using numeric attributes to encode the three performance metrics, we considered categorical data for our pattern recognition. Therefore, the dataset was discretised for further processing and the following three categories were considered as performance levels achievable by the three performance metrics:

$$1 \rightarrow LOW\ LEVELS\ of\ performance\ that\ is\ recommended$$
$$2 \rightarrow MODERATE\ LEVELS\ of\ performance\ that\ is\ acceptable\ but\ not\ ideal$$
$$3 \rightarrow HIGH\ LEVELS\ \ of\ performance\ that\ is\ not\ desired\ in\ the\ network$$

4.3.5.1. Noise

In the dataset, the noise performance was multiplied by 100 (for calibration purposes) and noise values were categorised using the following encoding:

$$If\ noise\ <\ -8500\ then\ encode \rightarrow 1\ (LOW)$$

$$Else\ if\ -8500\ <noise\ <\ -7000\ then\ encode \rightarrow 2\ (MODERATE)$$

$$Else\ if\ noise\ >-7000\ then\ encode \rightarrow 3\ (HIGH)$$

4.3.5.2. Load

The load expressed the UNIX kernel load average multiplied by 100. The load values were categorised using the following encoding:

$$If\ load\ <70\ then\ 1(LOW)$$

$$Else\ if\ 70\ <\ load<100\ then\ 2\ (MODERATE)$$

$$Else\ if\ load>100\ then\ 3\ (HIGH)$$

4.3.5.3. Uptime and Downtime

The Uptime is the device uptime considered as the current uptime (in minutes) since last reboot. The downtime rate per router is defined by:

$$\text{Downtime rate}\ = \frac{\text{number of reading with uptime} =< 60\ (indicating\ a\ device\ reset)}{\text{Total number of readings}}$$

The Downtime was categorised using the following encoding:

$$If\ \ downtime-rate\ >\ 1\%then\ 1\ (LOW)$$

$$Else\ if\ 5\% < downtime - rate < 1\%\ then\ 2\ (MODERATE)$$

$$Else\ if\ downtime - rate < 5\%\ then\ 3\ (HIGH)$$

4.3.5.4. Aberrant Performance Detection

Aberrant behaviour in performance can be seen as performance that deviates from normal or average performance. The standard deviation and statistical confidence bands can be used to measure aberrant behaviour in performance.

The empirical rule in statistics is that we classify aberrant behaviour as those observations that fall outside two standard deviation from the mean (approximately 95% of the standard confidence band); and those observations that fall outside three standard deviation from the mean (approximately 99.7$ of observations). The term we used in these experiments is the Delta (δ); and the Delta (δ) parameter will be classified as aberrant performance when it takes up values between 2 and 3 [70].

4.3.6. Test Cases

We conducted experiments using four test case scenarios revealing Wi-Fi operating constraints from loose (for example, rural setting where QoS is not an issue) to the most stringent (for example, suburban setting where modern applications demand QoS). In the experiments conducted, four sets of test cases were devised. These test cases are defined by Table 4.1 in terms of Noise, Load, Downtime rate (overall time the device was down) and the confidence band δ.

Table 4.1 Parameter Setting for Each Experiment Test Case

Test Cases	Noise(x100 dB)		Load (%)	Downtime (%)	Confidence Band
1	high	-7000	100	4%	1.00
	moderate	-8000	70		2.00
	low	-9000	0		3.00
2	high	-7250	100	3%	0.75
	moderate	-8250	75		1.75
	low	-9250	0		2.75
3	high	-7500	100	2%	0.50
	moderate	-8500	80		1.50
	low	-9500	0		2.50
4	high	-7750	100	1%	0.25
	moderate	-8750	85		1.25
	low	-9750	0		2.25

4.4. Experiment Results

Using the parameters described in the test cases of Table 4.1, we conducted another set of experiments to detect anomalous and aberrant network performance and different types of faults. Alongside good detectors revealing good performance, Table 4.2 below show a set of artificial immune non-self detectors that are anomalous and aberrant to the system's performance, i.e. not desired in the network. This table reflects a mapping from the encoding of performance levels of Table 4.1 into an antibody coding used by the AIS system.

Table 4.2: Non-self-Detectors for the Experiment

Antibody Code	Noise	Load	Uptime		Antibody Code	Noise	Load	Uptime
113	low	low	high		311	high	low	low
123	low	moderate	high		312	high	low	moderate
131	low	high	low		313	high	low	high
132	low	high	moderate		321	high	moderate	low
133	low	high	high		322	high	moderate	moderate
213	moderate	low	high		323	high	moderate	high
223	moderate	moderate	high		331	high	high	low
231	moderate	high	low		332	high	high	moderate
232	moderate	high	moderate		333	high	high	high
233	moderate	high	high					

The section that follows will reveal and discuss the results obtained from the experiments conducted in the Wi-Fi network monitoring research. It will discuss the network's anomaly performance then followed by the network's aberrant performance. This will include data-mining results for both anomaly and aberrant network performance.

4.4.1. Anomaly Performance

The objective of this study is to investigate whether there is a significant difference in average anomaly performance for the MLP, NaiveBayes and AIRS2 algorithms. To do this we will describe bar charts and employ hypothesis testing statistics. The study will discuss the results for Anomaly performance based on the following algorithm performance measures: Kappa Statistic, True Positive Rate, False Positive Rate, Precision, Recall, F-measure, ROC Area, and data mined by the three algorithms.

4.4.1.1. Anomaly Kappa Statistic Performance

The Kappa Statistic is used to measure the agreement between predicted and observed categorisation of a dataset, while correcting for agreement that occurs by chance.

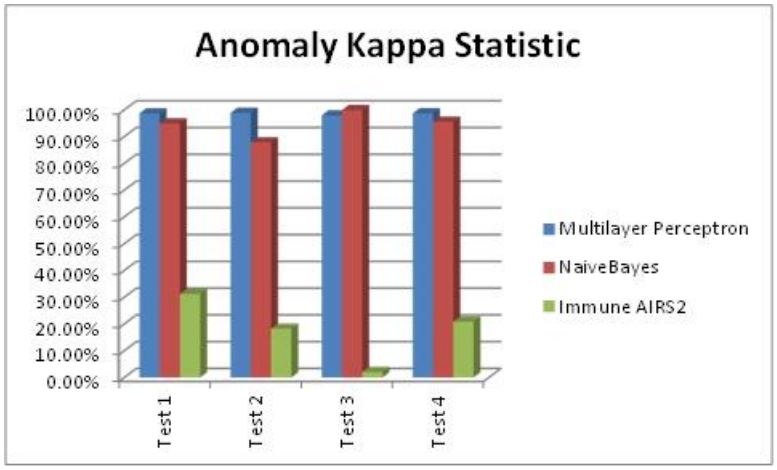

Figure 4.5: Anomaly Kappa Statistic Performance Bar Chart

Figure 4.5 above is a bar chart representation of the Kappa Statistic performance of the three algorithms across the four test cases. The MLP and NaiveBayes had a very good average Kappa Statistic performance of 98.59% and 94.60%, respectively. For MLP, test case 3 had the lowest Kappa statistic performance of 97.96% while the highest was 98.92% in test case 3. For NaiveBayes the lowest Kappa Statistic performance was 87.86% at test case 3 and the highest was 99.97% at test case 3.

The AIRS2 algorithm had a poor average Kappa Statistic Performance of 18.10%. In test case 1, the AIRS2's Kappa statistic performance was the highest at 31.19%, and the lowest was 2.07% in test case 3.

Table 4.3: Results for Anomaly Kappa Statistic T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.951177884	0.890611202
Variance	0.00050218	0.00515802
Observations	200	200
Pearson Correlation	0.318758675	
Hypothesized Mean Difference	0	
df	199	
t Stat	12.58238985	
P(T<=t) one-tail	2.14844E-27	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	4.29687E-27	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.951177884	0.1550969
Variance	0.00050218	0.015670599
Observations	200	200
Pearson Correlation	0.154521827	
Hypothesized Mean Difference	0	
df	199	
t Stat	91.00049877	
P(T<=t) one-tail	2.0705E-164	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	4.141E-164	
t Critical two-tail	1.971956544	

For the test:

Anomaly Kappa Statistic: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly Kappa Statistic: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 12.58 and its two-tailed p-value is 4.29687E-27. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude

that there is a difference in the mean anomaly Kappa Statistic for the MLP and NaiveBayes algorithms.

For the test:

Anomaly Kappa Statistic: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly Kappa Statistic: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 91.00 and its two-tailed p-value is 4.14E-164. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly Kappa Statistic for the MLP and AIRS2 algorithms.

4.4.1.2. Anomaly True Positive Rate Performance

True positive rate represents the rate at which a classifier algorithm fails to reject a true null hypothesis. This measures the accuracy of an algorithm to correctly classify.

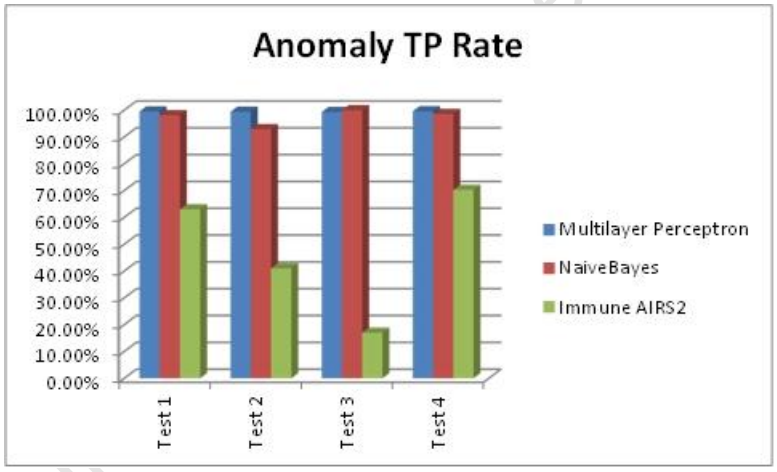

Figure 4.6: Anomaly True Positive Rate Performance Bar Chart

Figure 4.6 above is a bar chart representation of the Anomaly TP rate performance across the four test cases. The MLP and NaiveBayes had a very good average anomaly TP rate of 99.45% and 97.40%, respectively. For all test cases, the MLP algorithm had a TP Rate performance above 99.0%. The NaiveBayes algorithms achieved a 100% TP rate in test case 3 and the lowest being 92.90% in test case 2.

The AIRS2 algorithm had a poor anomaly TP rate performance of 47.83% across the test cases. In test case 4, the AIRS2 algorithm had a 70.30% TP rate and the lowest being 17.00% in test case 3.

Table 4.4: Results for Anomaly True Positive Rate T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.981019386	0.956329377
Variance	6.69234E-05	0.000737152
Observations	200	200
Pearson Correlation	0.381085825	
Hypothesized Mean Difference	0	
df	199	
t Stat	13.85865896	
P(T<=t) one-tail	2.5735E-31	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	5.147E-31	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.981019386	0.451863578
Variance	6.69234E-05	0.052385481
Observations	200	200
Pearson Correlation	0.259316282	
Hypothesized Mean Difference	0	
df	199	
t Stat	32.98175052	
P(T<=t) one-tail	6.70957E-83	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.34191E-82	
t Critical two-tail	1.971956544	

For the test:

Anomaly TP Rate: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly TP Rate: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 13.858 and its two-tailed p-value is 5.147E-31. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly True Positive Rate for the MLP and NaiveBayes algorithms.

For the test:

Anomaly TP Rate: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly TP Rate: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 32.981 and its two-tailed p-value is 1.34191E-82. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly True Positive Rate for the MLP and AIRS2 algorithms.

4.4.1.3. Anomaly False Positive Rate Performance

The false positive rate represents the rate at which a classifier algorithm will reject a true null hypothesis. This measures the accuracy of an algorithm to correctly classify.

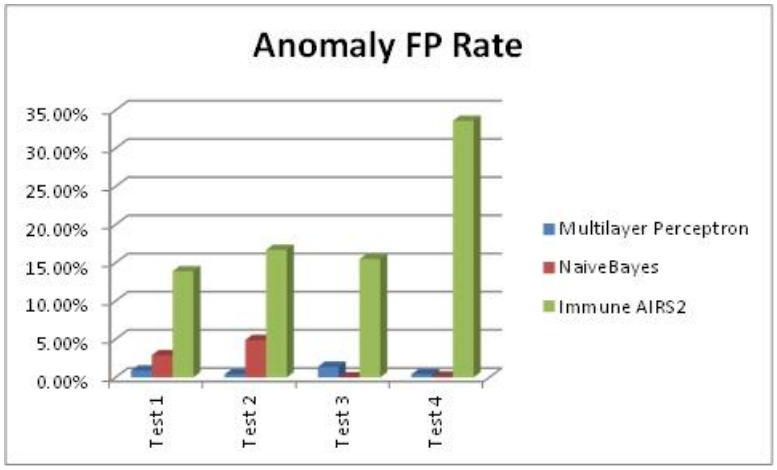

Figure 4.7: Anomaly False Positive Rate Performance Bar Chart

The bar chart representation in Figure 4.7 reveals that across all four test cases, the MLP and NaiveBayes had a very good average FP rate of 0.78% and 1.98%, respectively.

The AIRS2 algorithm had a much higher average FP rate performance of 19.93% across all test cases. The AIRS2 algorithm had the higher anomaly FP rate of 33.60% in test case 4.

Table 4.5: Results for Anomaly False Positive Rate T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.023073725	0.030094366
Variance	0.000857192	0.001199433
Observations	200	200
Pearson Correlation	-0.001087932	
Hypothesized Mean Difference	0	
df	199	
t Stat	-2.188171447	
P(T<=t) one-tail	0.014910774	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	0.029821549	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.023073725	0.350713599
Variance	0.000857192	0.040716876
Observations	200	200
Pearson Correlation	-0.076387248	
Hypothesized Mean Difference	0	
df	199	
t Stat	-22.4820919	
P(T<=t) one-tail	7.89418E-57	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.57884E-56	
t Critical two-tail	1.971956544	

For the test:

Anomaly FP Rate: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly FP Rate: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -2.188 and its two-tailed p-value is 0.0298. At 5% confidence level, the test is significant and there is strong evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly False Positive Rate for the MLP and NaiveBayes algorithms.

For the test:

Anomaly FP Rate: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly FP Rate: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is -22.482 and its two-tailed p-value is 1.57884E-56. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly False Positive Rate for the MLP and AIRS2 algorithms.

4.4.1.4. Anomaly Precision Performance

A classification algorithm's Precision measures the percentage of instances classified as 'positive' were indeed 'positive'.

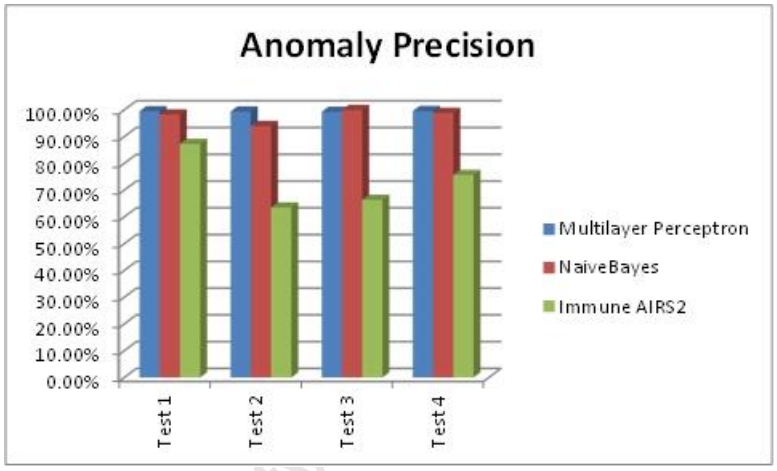

Figure 4.8: Anomaly Precision Performance Bar Chart

Figure 4.8 above is a bar chart representation of results for the four test cases. The chart reveals that the MLP had an average Precision of 99.45%, while the NaiveBayes had an average Precision rate of 97.80% across the four test cases. In all test cases, The MLP had anomaly precision above 99.0%. The NaiveBayes had the highest precision of 100% in test case 3 and the lowest being 94.0% in test case 2.

The AIRS2 algorithm had a satisfactory precision of 73.23%. The AIRS2 algorithm had the lowest precision performance of 63.50% in test case 2 and the highest being 87.30% in test case 1.

Table 4.6: Results for Anomaly Precision T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.981026481	0.962379322
Variance	5.82305E-05	0.000569478
Observations	200	200
Pearson Correlation	0.267201097	
Hypothesized Mean Difference	0	
df	199	
t Stat	11.45060946	
P(T<=t) one-tail	5.98026E-24	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.19605E-23	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.981026481	0.691778549
Variance	5.82305E-05	0.026351371
Observations	200	200
Pearson Correlation	0.024123976	
Hypothesized Mean Difference	0	
df	199	
t Stat	25.1997544	
P(T<=t) one-tail	3.87155E-64	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	7.74311E-64	
t Critical two-tail	1.971956544	

For the test:

Anomaly Precision: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly Precision: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 11.450 and its two-tailed p-value is 1.19605E-23. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly precision for the MLP and NaiveBayes algorithms.

For the test:

Anomaly Precision: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly Precision: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 25.199 and its two-tailed p-value is 7.74311E-64. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly precision for the MLP and AIRS2 algorithms.

4.4.1.5. Anomaly Recall Performance

A classification algorithm's Recall measures how well it can recognise positive instances and samples.

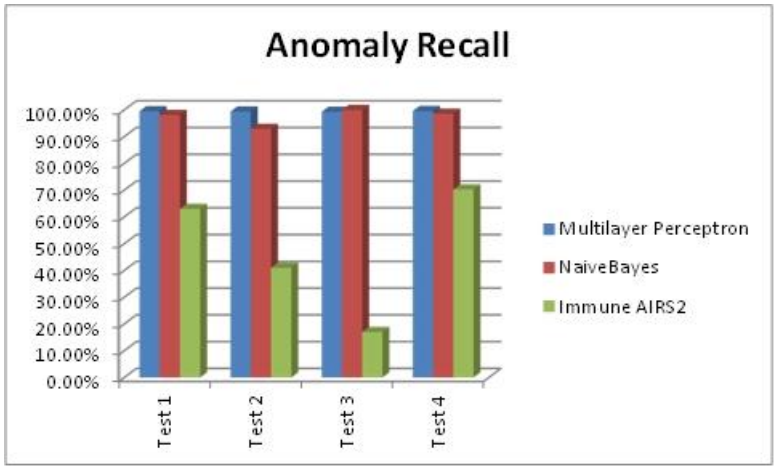

Figure 4.9: Anomaly Recall Performance Bar Chart

Figure 4.9 above is a bar chart representation of the Anomaly Recall performance for the three algorithms across four test cases. The MLP and NaiveBayes had an average anomaly recall of 99.45% and 97.40%, respectively. In test case 3, the NaiveBayes achieved a 100% Anomaly recall, displaying a good performance above the MLP. The AIRS2 had a poor average anomaly recall of 47.83%.

Table 4.7: Results for Anomaly Recall T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.981019386	0.956329377
Variance	6.69234E-05	0.000737152
Observations	200	200
Pearson Correlation	0.381085825	
Hypothesized Mean Difference	0	
df	199	
t Stat	13.85865896	
P(T<=t) one-tail	2.5735E-31	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	5.147E-31	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.981019386	0.451863578
Variance	6.69234E-05	0.052385481
Observations	200	200
Pearson Correlation	0.259316282	
Hypothesized Mean Difference	0	
df	199	
t Stat	32.98175052	
P(T<=t) one-tail	6.70957E-83	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.34191E-82	
t Critical two-tail	1.971956544	

For the test:

Anomaly Recall: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly Recall: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 13.858 and its two-tailed p-value is 5.147E-31. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly recall for the MLP and NaiveBayes algorithms.

For the test:

Anomaly Recall: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly Recall: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 32.981 and its two-tailed p-value is 1.34191E-82. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly recall for the MLP and AIRS2 algorithms.

4.4.1.6. Anomaly F-measure Performance

There is a trade-off between Precision (P) and Recall (R) measures. When one tries to improve the first measure, there is often deterioration in the second measure. This represents a Multi-Criteria-Decision-Making (MCDM) problem, and the F-measure provides a harmonic mean for the MCDM problem.

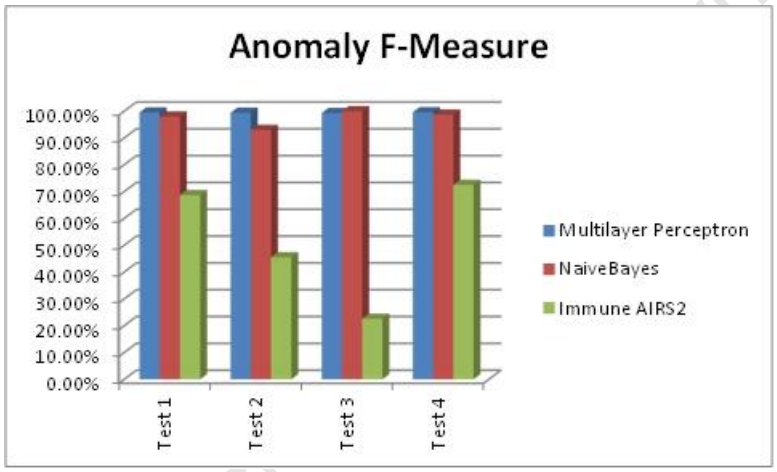

Figure 4.10: Anomaly F-measure Performance Bar Chart

Figure 4.10 above reveals that the MLP and NaiveBayes had an average anomaly F-measure of 99.45% and 97.43%, respectively, across the four test cases. The AIRS2 algorithm had an average anomaly F-measure of 52.33%. The AIRS2 had the lowest F-measure in test case 3 and the highest F-measure performance in test case 4.

Table 4.8: Results for Anomaly F-Measure T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.980355749	0.952768166
Variance	7.10445E-05	0.001012232
Observations	200	200
Pearson Correlation	0.302042597	
Hypothesized Mean Difference	0	
df	199	
t Stat	12.85383371	
P(T<=t) one-tail	3.16288E-28	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	6.32577E-28	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.980355749	0.485149115
Variance	7.10445E-05	0.048150873
Observations	200	200
Pearson Correlation	0.218983518	
Hypothesized Mean Difference	0	
df	199	
t Stat	32.16309945	
P(T<=t) one-tail	4.54456E-81	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	9.08911E-81	
t Critical two-tail	1.971956544	

For the test:

Anomaly F-Measure: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly F-Measure: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 12.853 and its two-tailed p-value is 6.32577E-28. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly F-measure for the MLP and NaiveBayes algorithms.

For the test:

Anomaly F-Measure: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly F-Measure: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 32.163 and its two-tailed p-value is 9.08911E-81. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly F-measure for the MLP and AIRS2 algorithms.

4.4.1.7. Anomaly ROC Area Performance

Receiver Operating Characteristics (ROC) curves represents a trade-off between true positive rate and false positive rate. The area under the curve measures the probability that a classification algorithm will rank a randomly chosen positive instance above a randomly chosen negative one.

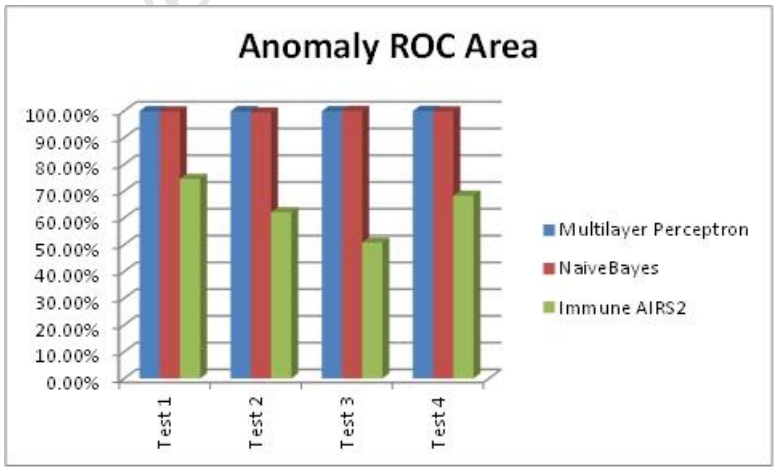

Figure 4.11: Anomaly ROC Area Performance Bar Chart

The bar chart in Figure 4.11 above reveals that the MLP and NaiveBayes had a satisfactory ROC Area performance; they had an average ROC area performance of 99.93% and 99.75%,

respectively, across the four test cases. The AIRS2 had an average anomaly ROC Area performance of 63.95% across all test cases.

Table 4.9: Results for Anomaly ROC Area T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.99883819	0.993997907
Variance	9.9612E-07	2.59475E-05
Observations	200	200
Pearson Correlation	0.781877817	
Hypothesized Mean Difference	0	
df	199	
t Stat	15.70661488	
P(T<=t) one-tail	5.42009E-37	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.08402E-36	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.99883819	0.610922711
Variance	9.9612E-07	0.009293489
Observations	200	200
Pearson Correlation	0.583693195	
Hypothesized Mean Difference	0	
df	199	
t Stat	57.25053863	
P(T<=t) one-tail	7.1498E-126	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.43E-125	
t Critical two-tail	1.971956544	

For the test:

Anomaly ROC Area: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly ROC Area: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 15.706 and its two-tailed p-value is 1.08402E-36. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly ROC Area for the MLP and NaiveBayes algorithms.

For the test:

Anomaly ROC Area: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly ROC Area: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 57.250 and its two-tailed p-value is 1.43E-125. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly ROC Area for the MLP and AIRS2 algorithms.

4.4.2. Aberrant Performance

Aberrant performance represents behaviour in a network that is noticed over a longer period of time, performance that deviates from normal or known performance. This kind of behaviour was detected in the network and we used the three algorithms to research their ability to detect and correctly classify this kind of network performance. To measure their ability to correctly classify, we test them against the following algorithm performance

measures: Kappa Statistic, True Positive Rate, False Positive Rate, Precision, Recall, F-measure, ROC Area, and data mined by the three algorithms.

4.4.2.1. Aberrant Kappa Statistic Performance

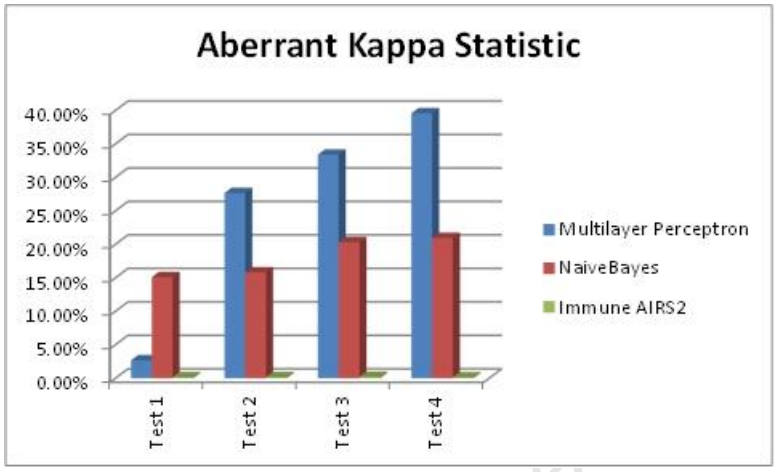

Figure 4.12: Aberrant Kappa Statistic Performance Bar Chart

Figure 4.12 above displays a bar chart representation of the Aberrant Kappa Statistic performance from the experiments. The MLP and NaiveBayes had a poor average aberrant Kappa statistic of 25.83% and 18.05%. In test case 1 the MLP had a 2.68% Kappa statistic performance; this kept increasing until 39.57% in test case 4. The NaiveBayes had a Kappa statistic performance between 15% and 20% across all test cases.

The AIRS2 had an even worse average aberrant Kappa statistic of 0.16% across all test cases.

Table 4.10: Results for Aberrant Kappa Statistic T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.023643456	0.18131208
Variance	0.005530488	0.001301372
Observations	200	200
Pearson Correlation	-0.074721628	
Hypothesized Mean Difference	0	
df	199	
t Stat	-26.21848711	
P(T<=t) one-tail	8.94959E-67	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.78992E-66	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.023643456	0.001641817
Variance	0.005530488	6.85002E-06
Observations	200	200
Pearson Correlation	0.06337119	
Hypothesized Mean Difference	0	
df	199	
t Stat	4.190724142	
P(T<=t) one-tail	2.08977E-05	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	4.17955E-05	
t Critical two-tail	1.971956544	

For the test:

Aberrant Kappa Statistic: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant Kappa Statistic: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -26.2185 and its two-tailed p-value is 1.78992E-66. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant Kappa statistic for the MLP and NaiveBayes algorithms.

For the test:

Aberrant Kappa Statistic: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant Kappa Statistic: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 4.191 and its two-tailed p-value is 4.17955E-5. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant Kappa statistic for the MLP and AIRS2 algorithms.

4.4.2.2. Aberrant True Positive Rate Performance

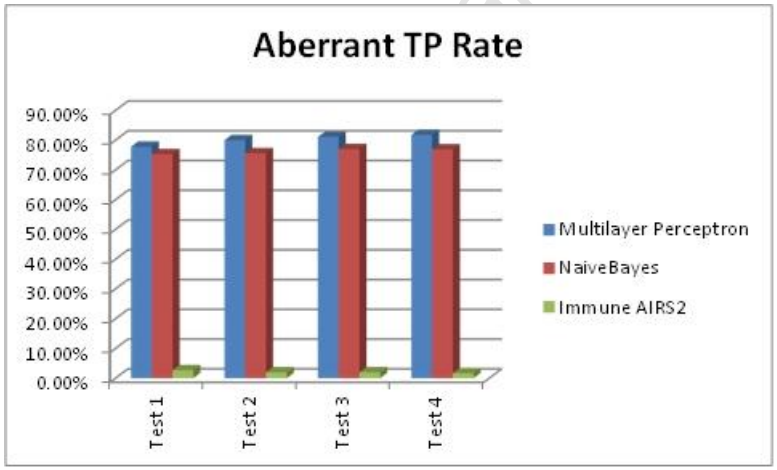

Figure 4.13: Aberrant True Positive Rate Performance Bar Chart

The bar chart in Figure 4.13 depicts that MLP and NaiveBayes had a fair aberrant TP rate performance. The average TP rate performance for MLP and NaiveBayes was at 80.18% and 76.20%, respectively, across all test cases. The MLP had its highest performance of 81.80% in test case 4, and its lowest performance of 77.80% in test case 1. The NaiveBayes had a TP rate performance between 75.30% and 77.00%. The AIRS2 had a poor average aberrant TP rate performance of 2.10%.

Table 4.11: Results for Aberrant True Positive Rate T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.777601976	0.762373199
Variance	5.53854E-05	9.25227E-05
Observations	200	200
Pearson Correlation	-0.350717259	
Hypothesized Mean Difference	0	
df	199	
t Stat	15.30084774	
P(T<=t) one-tail	9.47268E-36	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.89454E-35	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.777601976	0.020179732
Variance	5.53854E-05	0.000557869
Observations	200	200
Pearson Correlation	0.115493083	
Hypothesized Mean Difference	0	
df	199	
t Stat	447.6180056	
P(T<=t) one-tail	4.1071E-301	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	8.2142E-301	
t Critical two-tail	1.971956544	

For the test:

Aberrant True Positive Rate: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant True Positive Rate: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 15.300 and its two-tailed p-value is 1.89454E-35. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant true positive rate for the MLP and NaiveBayes algorithms.

For the test:

Aberrant True Positive Rate: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant True Positive Rate: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 447.618 and its two-tailed p-value is 8.2142E-301. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant true positive rate for the MLP and AIRS2 algorithms.

4.4.2.3. Aberrant False Positive Rate Performance

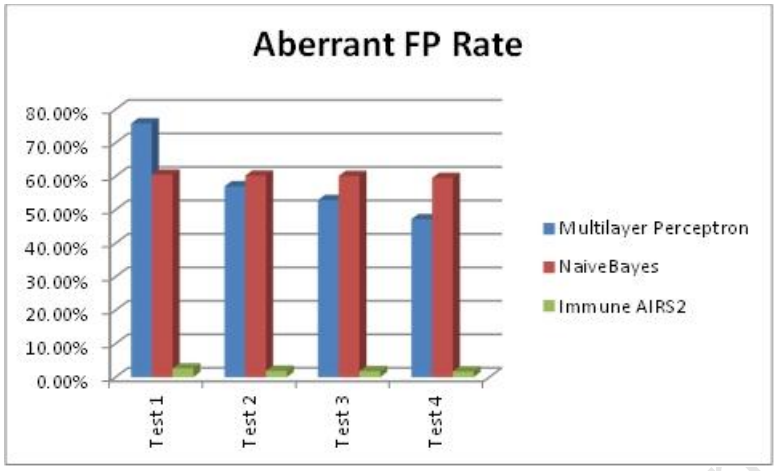

Figure 4.14: Aberrant False Positive Rate Performance Bar Chart

The MLP and NaiveBayes had a very high and poor average aberrant FP rate of 58.23% and 60.08%, respectively. The AIRS had a very good average aberrant FP rate of 2.00% across all test cases. This is shown by the bar chart graph of Figure 4.14.

Table 4.12: Results for Aberrant False Positive Rate T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.758605437	0.60128009
Variance	0.002883446	0.000595701
Observations	200	200
Pearson Correlation	0.362615122	
Hypothesized Mean Difference	0	
df	199	
t Stat	44.2454333	
P(T<=t) one-tail	3.136E-105	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	6.272E-105	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.758605437	0.018513939
Variance	0.002883446	0.00041616
Observations	200	200
Pearson Correlation	-0.085337631	
Hypothesized Mean Difference	0	
df	199	
t Stat	177.256146	
P(T<=t) one-tail	2.7777E-221	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	5.5555E-221	
t Critical two-tail	1.971956544	

For the test:

Aberrant False Positive Rate: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant False Positive Rate: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 44.245 and its two-tailed p-value is 6.272E-105. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant false positive rate for the MLP and NaiveBayes algorithms.

For the test:

Aberrant False Positive Rate: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant False Positive Rate: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 177.256 and its two-tailed p-value is 5.5555E-221. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant false positive rate for the MLP and AIRS2 algorithms.

4.4.2.4. Aberrant Precision Performance

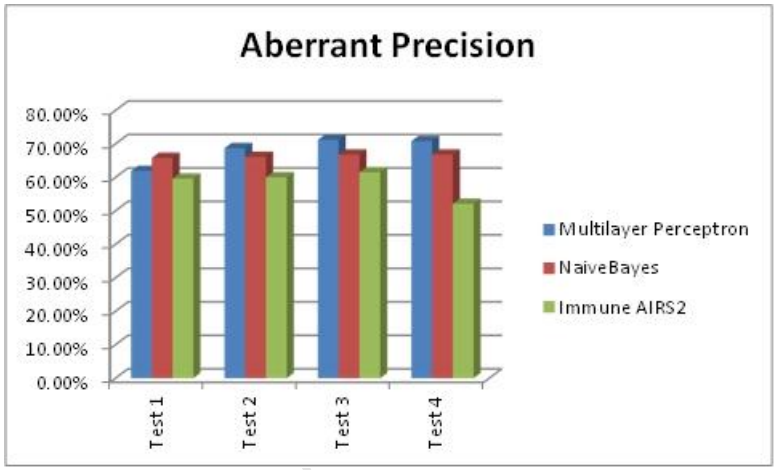

Figure 4.15: Aberrant Precision Performance Bar Chart

The three algorithms had a fair aberrant precision performance. The MLP and NaiveBayes had an average aberrant precision performance of 68.15% and 66.38%, respectively. The AIRS2 had an average aberrant precision performance of 58.30%. This is shown by the bar chart graph of figure 4.15. For all the three algorithms, their performance was almost similar and fluctuated between 59% and 71%. The MLP performance kept increasing from 61.90% in test case 1 to the highest 71.20% in test case 3. A similar increasing performance was displayed by the NaiveBayes by improving from a 65.80% in test case 1 to 66.80% in test case 4. The opposite performance was then displayed by the NaiveBayes with a 59.60% Precision performance in test case 1 until it decreased to a 52.10% in test case 4.

Table 4.13: Results for Aberrant Precision T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.613185772	0.661064761
Variance	0.000401774	5.59138E-05
Observations	200	200
Pearson Correlation	0.251188485	
Hypothesized Mean Difference	0	
df	199	
t Stat	-34.62635234	
P(T<=t) one-tail	1.74714E-86	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	3.49428E-86	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.613185772	0.156895698
Variance	0.000401774	0.067678827
Observations	200	200
Pearson Correlation	-0.06186818	
Hypothesized Mean Difference	0	
df	199	
t Stat	24.6147836	
P(T<=t) one-tail	1.34008E-62	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	2.68015E-62	
t Critical two-tail	1.971956544	

For the test:

Aberrant Precision: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant Precision: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -34.626 and its two-tailed p-value is 3.49428E-86. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant precision for the MLP and NaiveBayes algorithms.

For the test:

Aberrant Precision: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant Precision: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 24.614 and its two-tailed p-value is 2.68015E-62. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant precision for the MLP and AIRS2 algorithms.

4.4.2.5. Aberrant Recall Performance

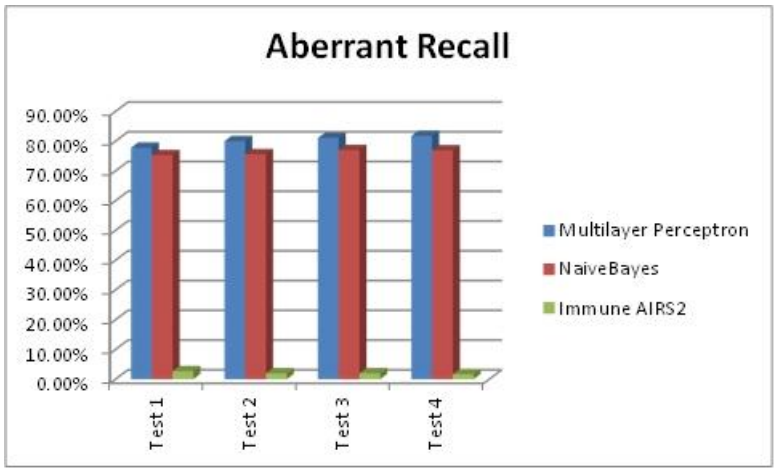

Figure 4.16: Aberrant Recall Performance Bar Chart

The bar chart in figure 4.16 reveals that the MLP had an average aberrant Recall performance of 80.18%, while the NaiveBayes had an average aberrant performance of 76.20% across all test cases. The AIRS2 algorithm had a poor average aberrant recall of 2.10% across all test cases. The MLP had a 77.80% Recall performance in test case 1; this kept improving until an 81.80% performance in test case 4. The NaiveBayes had a similar trend; it started with a 75.30% performance in test case 1, and improved to 76.90% in test case 4.

Table 4.14: Results for Aberrant Recall T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.777601976	0.762373199
Variance	5.53854E-05	9.25227E-05
Observations	200	200
Pearson Correlation	-0.350717259	
Hypothesized Mean Difference	0	
df	199	
t Stat	15.30084774	
P(T<=t) one-tail	9.47268E-36	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.89454E-35	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.777601976	0.020179732
Variance	5.53854E-05	0.000557869
Observations	200	200
Pearson Correlation	0.115493083	
Hypothesized Mean Difference	0	
df	199	
t Stat	447.6180056	
P(T<=t) one-tail	4.1071E-301	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	8.2142E-301	
t Critical two-tail	1.971956544	

For the test:

Aberrant Recall: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant Recall: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 15.300 and its two-tailed p-value is 1.89454E-35. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant recall for the MLP and NaiveBayes algorithms.

For the test:

Aberrant Recall: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant Recall: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 447.618 and its two-tailed p-value is 8.2142E-301. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant recall for the MLP and AIRS2 algorithms.

4.4.2.6. Aberrant F-measure Performance

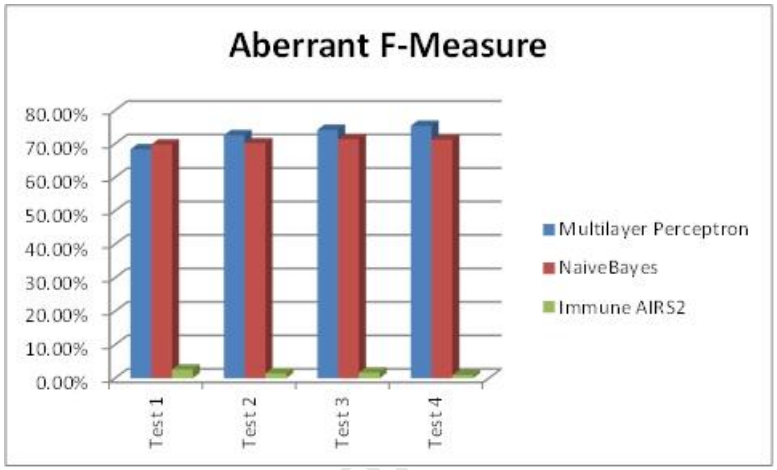

Figure 4.17: Aberrant F-measure Performance

Figure 4.17 above displays a bar chart graph of results, and this indicates that, across all test cases, the MLP and NaiveBayes algorithms had an average aberrant f-measure performance of 72.65% and 70.65%, respectively. On the other hand, the AIRS2 algorithms had poor average aberrant F-measure performance of 1.68%. The MLP had an F-measure performance of 68.40% in test case 1, this kept improving until a 75.40% in test case 4. A similar trend in performance can be noticed with the NaiveBayes algorithm; in test case 1, the F-measure performance was 69.80% and this kept slightly improving until a 71.20% performance in test case 4.

Table 4.15: Results for Aberrant F-measure T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.68125069	0.705628234
Variance	0.000162554	7.53478E-05
Observations	200	200
Pearson Correlation	-0.192488573	
Hypothesized Mean Difference	0	
df	199	
t Stat	-20.58413054	
P(T<=t) one-tail	1.73072E-51	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	3.46143E-51	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.68125069	0.014703114
Variance	0.000162554	0.001265689
Observations	200	200
Pearson Correlation	0.107304574	
Hypothesized Mean Difference	0	
df	199	
t Stat	258.3887656	
P(T<=t) one-tail	1.0385E-253	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	2.077E-253	
t Critical two-tail	1.971956544	

For the test:

Aberrant F-measure: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant F-measure: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -20.584 and its two-tailed p-value is 3.46143E-51. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant F-measure for the MLP and NaiveBayes algorithms.

For the test:

Aberrant F-measure: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant F-measure: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 258.388 and its two-tailed p-value is 2.077E-253. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant F-measure for the MLP and AIRS2 algorithms.

4.4.2.7. Aberrant ROC Area Performance

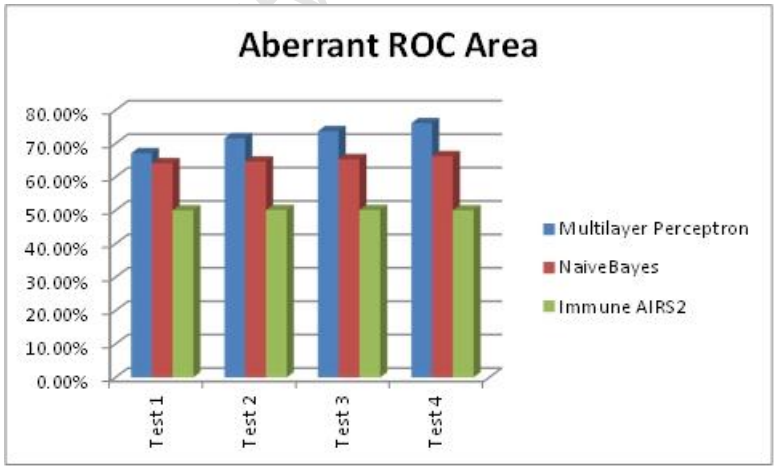

Figure 4.18: Aberrant ROC Area Performance

Across all test cases, the MLP and NaiveBayes had an average aberrant ROC area of 72.13% and 65.05%, respectively. The AIRS2 had an average aberrant ROC Area of 50.05%. This is displayed in figure 4.18's bar chart graph.

Table 4.16: Results for Aberrant ROC Area T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.60278162	0.646713804
Variance	0.002336787	0.000853362
Observations	200	200
Pearson Correlation	0.873556823	
Hypothesized Mean Difference	0	
df	199	
t Stat	-23.10639025	
P(T<=t) one-tail	1.52247E-58	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	3.04494E-58	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.60278162	0.500832897
Variance	0.002336787	6.62684E-06
Observations	200	200
Pearson Correlation	0.112713051	
Hypothesized Mean Difference	0	
df	199	
t Stat	29.96314476	
P(T<=t) one-tail	5.46421E-76	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.09284E-75	
t Critical two-tail	1.971956544	

For the test:

Aberrant ROC Area: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant ROC Area: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -23.106 and its two-tailed p-value is 3.04494E-58. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant ROC Area for the MLP and NaiveBayes algorithms.

For the test:

Aberrant ROC Area: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant ROC Area: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 29.963 and its two-tailed p-value is 1.09284E-75. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant ROC Area for the MLP and AIRS2 algorithms.

4.4.3. Data Mining Using a Confusion Matrix

This section of the results chapter will reveal the data mined using the algorithms, for both anomaly and aberrant performance detection. The section will discuss results from the anomaly performance detection, followed by aberrant performance detection.

4.4.3.1. Anomaly Performance Detection

Anomaly Performance is network performance that is above a given threshold, one that is unacceptable in the network. This section will discuss results obtained from anomaly detection using the three algorithms.

4.4.3.1.1. Multilayer Perceptron Anomaly Detection

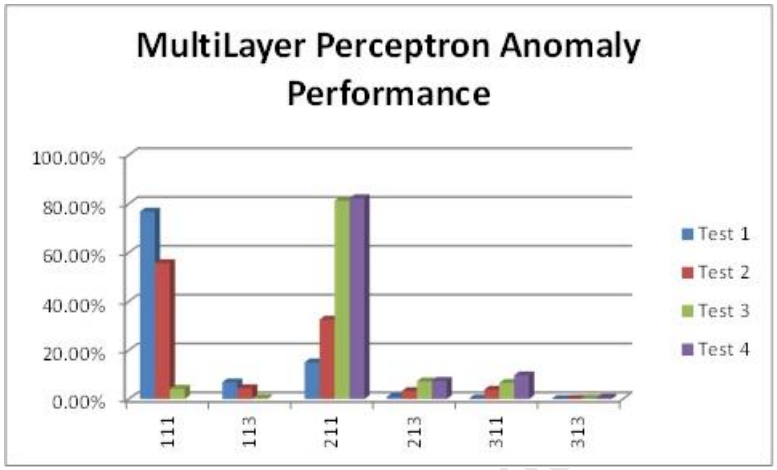

Figure 4.19: Multilayer Perceptron Anomaly Detection Bar Chart

The MLP identified 6 types of detectors. Among those detectors that were identified, the '111' and '211' detectors represent good and desired network performance. The '111' represents good noise, load and downtime performance, while '211' represents moderate noise levels with low load and low downtime performance. These detectors contributed 76.78% to 82.28% of the network's anomaly performance.

From Test cases 1 to 4, the detection rate of '111' kept decreasing whilst that of '211' increased. This is an indication that as the network thresholds are made more stringent, a large portion of '111' detectors became '211'. This indicates a good network performance.

Not only did the '111' type performance change to '211' performance when thresholds were made more stringent. Changing parameter settings from test case 1 to test case 4 brought about an increase in '311' type performance. The '311' type performance represents bad noise performance, with low load levels and low downtime levels. This can be correlated with the change in detector rate from '111' to '211' and '311'. They '311' detector type contributed 0.28% to 9.72% of the network's performance throughout test cases 1 to 4.

Another detector identified in the network was that of type '113', the detector represents those devices in the network experiencing high levels of downtime, with low noise and low load. As the performance thresholds are made more stringent from test case 1 to 4, its detection rate decrease from 6.86% to 0.39%.

From test case 1 to 4, there was a decrease of '113' detectors, this can be correlated with an increase of type '213' and '313' detectors. Type '213' represents those devices in the

network that had experienced moderate levels of noise with low load level, but with high levels of downtime. The '313' detectors represents those devices in the network that experienced high levels of noise and downtime, with very low levels of load performance. The '213' increased from 0.99% in test case1, to 7.46% in test case 4. A similar increasing detection rate was shown by the '313' detector when it increased from 0.04% in test case 1 to 0.54% in test case 4.

4.4.3.1.2. NaiveBayes Anomaly Detection

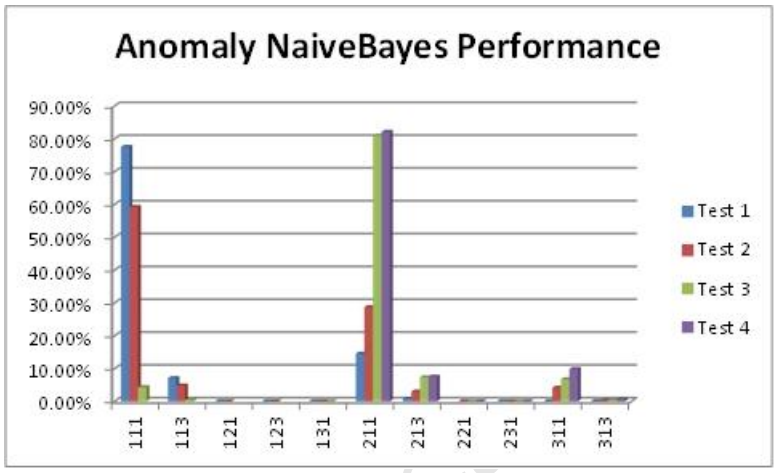

Figure 4.20: NaiveBayes Anomaly Detection Bar Chart

The NaiveBayes algorithm was able to identify 11 types of detectors. Although the NaiveBayes had identified 11 detectors in comparison with the 6 detectors identified by the MLP, the two algorithms had a similar performance.

From test case 1 to test case 4, the detection rate of the '111' detector decreased from a high 77.53% to a low 4.32%. On the other hand, as the detection rate of the '111' detector decreased; there was an increase in detection rate for the '211' and '311' type detectors.

The detection rate of '211' increased from a low 14.53% in test case 1 to a high 82.10% in test case 4. A similar increase was seen when the '311' detection rate increased from 0.02% in test case 1 to 9.84% in test case 4.

The detection rate of the '113' detector in test case 1 was 7.08%, and this decreased to 4.32% in test case 3, and 0% detection rate in test case 4. The decrease of '113' detection rate can be correlated with the increase in '213' and '313' detector types.

The detection rate of '213' increased from 0.82% in test case 1, to 7.51% in test case 4. A similar performance was experienced with the '313' type detector. Its detection rate increased from 0.03% in test case 1 to 0.55% in test case 4.

The NaiveBayes was able to identify 5 more detectors, the '121', 123', 131', 221', and the 231'. This group of detectors each contributed to less than 0.01% performance detection rate in the network.

4.4.3.1.3. AIRS2 Anomaly Detection

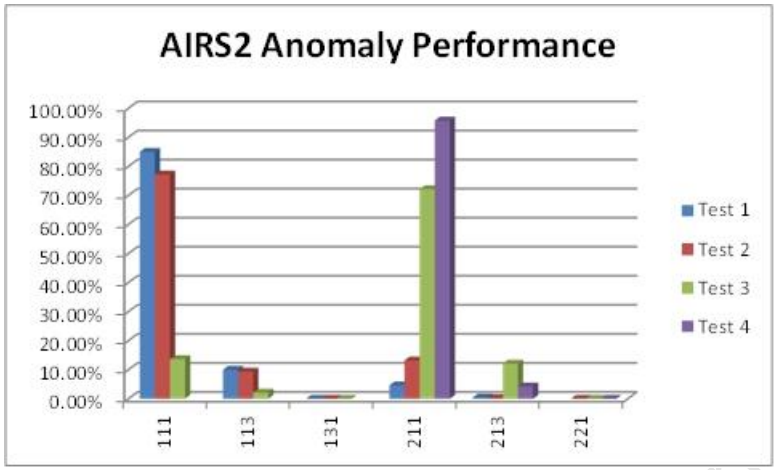

Figure 4.21: AIRS2 Anomaly Detection Bar Chart

The AIRS2 algorithm was able to identify 6 detectors in the network and those that dominated the network were the '111' and the '211' type detectors. These detectors represent good performance of the network. The detection rate for the '111' type detector was 84.99% in test case 1, and this was reduced to 13.64% in test case 3 and 0% in test case 4. This can be related to an increased detection rate of '211' type detectors, in test case 1 its detection rate was 4.61% and this increased to 95.77% in test case 4.

The bar chart in figure 4.21 indicates that in test case 1, the '113' detection rate was 10.00%, and this was reduced to 2.14% in test case 3 and to 0% in test case 4. This decrease can be related with the increased detection rate of '213'. In test case 1, the detection rate of '213' was 0.39%, and this increased to 12.09% in test case 3 and 4.23% in test case 4.

The detectors that had low detection rate were that of type '131' and '221'. They both had a detection rate less than 0.01% for all test cases 1 to 4. '131' represents devices that had experienced high levels of load, with low levels of both noise and downtime. The '221' detector represent devices with moderate noise and load performance, with low downtime performance.

4.4.3.2. Aberrant Performance Detection

Aberrant performance represents those devices in the network that has experienced high and unstable changes in noise, load or downtime performance. Detecting and remedying this kind of performance can lead to pre-emptive measures in preventing network failure in the future.

This section will reveal and discuss results from the aberrant performance experiments for each algorithm.

4.4.3.2.1. Multilayer Perceptron Aberrant Detection

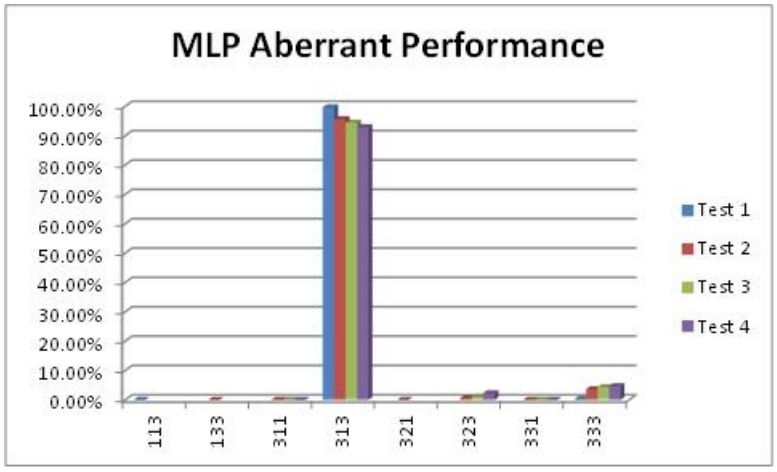

Figure 4.22: MLP Aberrant Detection Bar Chart

The MLP was able to identify 8 detectors. The most dominant was detector type '313', and it represents those devices in the network that have experienced prolonged, unstable and high changes in noise and downtime performance with low changes in noise levels. The '313' had a very high detection rate, 99.70% in test case 1, and this detection rate kept dropping until 92.95% in test case 4.

The decrease in the detection rate of the '313' detector can be linked to an increase in detection rate of the '323' and '333' type detectors.

The '323' represents those devices in the network that have experienced prolonged, unstable and high changes in noise and downtime performance with moderate changes in noise levels. The detection rate of '323' increased from 0% in test case 1, to 2.31% in test case 4.

The '333' represents those devices in the network that have experienced prolonged, unstable and high changes in noise, load and downtime performance. This is a state of the device that is not desirable by a network administrator. Its detection rate increased from 0.30% in test case 1, to 4.73% in test case 4.

The detection rate for '113', '133', '311' and '321' was very low at 0.05%.

4.4.3.2.2. NaiveBayes Aberrant Detection

The NaiveBayes was able to identify 8 detectors. These are the same detectors as those identified in the MLP Aberrant Detection experiment.

The most pre-eminent detector was the type '313', its detection rate in test case 1 was 97.62% and this kept decreasing to 95.66% in test case 4. This decrease in detection rate can be correlated to an increase in the detector type '333'.

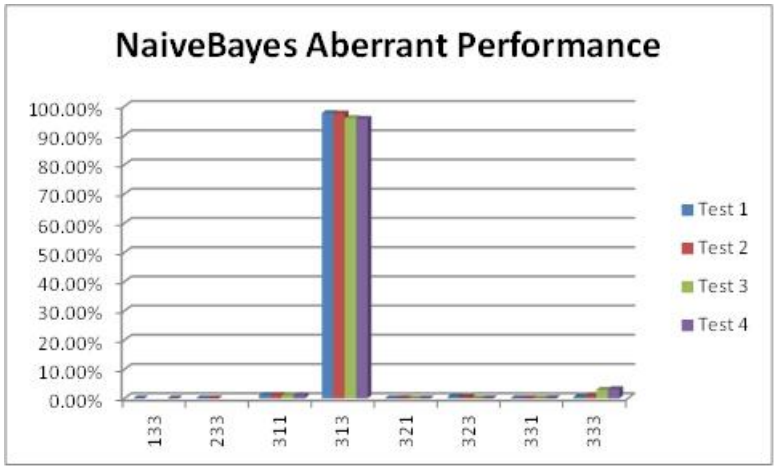

Figure 4.23: NaiveBayes Aberrant Detection Bar Chart

For the NaiveBayes experiment, the detection rate of the '323' detector was very different from that of MLP. Its detection rate decreased in NaiveBayes experiment, as opposed to an increased detection rate in the MLP experiment. In test case 1, the detection rate was at 0.62%, and this decreased to a 0.02% detection rate in test case 4.

In test case 1, the detection rate of '333' was 0.54% and this kept increasing until 3.16% in test case 4. This is a similar performance compared to the MLP.

4.4.3.2.3. AIRS2 Aberrant Detection

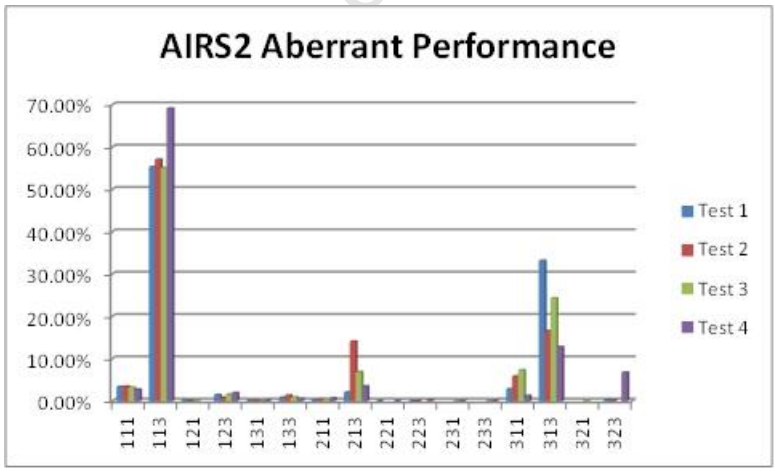

Figure 4.24: AIRS2 Aberrant Detection Bar Chart

The AIRS2 algorithm was able to identify 16 detectors. The most pre-eminent were the '113', '213', '311' and '313' type detectors.

The '113' detector represents those network devices experiencing unstable and high changes in downtime performance, with moderate changes in both noise and load. Its detection rate in test case 1 was 55.28% and this kept increasing until 69.12% in test case 4.

The ‘213’ type detector represents those network devices that had experienced unstable and high changes in downtime level, with moderate noise and low load performance. In test case 1, its detection rate was 2.06% and in test case 2 it got to its highest of 14.05%. The detection rate dropped to 3.49% in test case 4.

The ‘311’ represents those devices in the network that experienced unstable and high changes in noise performance, with low changes in both load and downtime performance. Its detection rate in test case 1 was 2.83%, this peaked to 7.29% in test case 3 and it dropped to a 1.24% detection rate in test case 4.

The ‘313’ represents those devices in the network that had experienced unstable and high changes in noise and downtime performance, but with a low changes in load performance. Its detection rate reached the highest peak in test case 1, but this was reduced to a 12.77% in test case 4.

With the AIRS2 the detector types ‘111’, ‘121’, ‘123’, ‘131’, ‘133’, ‘211’, ‘221’, ‘223’, ‘231’, ‘233’, ‘321’ and ‘313’ had an average detection rate of less than 5.00% and in some cases 0%.

4.5. Conclusion

4.5.1. Anomaly Network Performance Detection

The statistical hypothesis test experiments conducted for anomaly performance detection reveal that in all eight algorithm performance measures there is a significant mean difference among the three algorithms.

The bar chart representations in figure 4.5 to 4.11 were carefully examined, and for all performance measures, the MLP has had an overall good performance and came out with the highest (above90%) algorithm performance measures. The NaiveBayes also had a good performance that was slightly lower than that of the MLP; it came second after the MLP. On the other hand, the AIRS2 had a poor performance relative to the MLP and NaiveBayes.

The data-mining capabilities of the MLP and NaiveBayes were statistically different, but the NaiveBayes identified the same detectors as the MLP, and even more detectors that the MLP was unable to identify. The detection rate of those common detectors was slightly similar.

The AIRS2 identified a few common detectors to both the MLP and NaiveBayes, with differing average detection rates to those of the MLP and NaiveBayes. The accuracy of the AIRS2 was poor, at 60% average; therefore one can question the accuracy of the data mining evaluations revealed in section 4.4.3 of the results section.

Overall, the MLP showed excellent accuracy in network anomaly performance detection. The NaiveBayes came second while the AIRS2 had poor network anomaly performance detection.

4.5.2. Aberrant Network Performance Detection

As shown in section 4.2, the statistical hypothesis test experiments conducted for network aberrant performance detection indicate that there is a statistical difference in mean performance measures for the MLP, NaiveBayes and AIRS2 in all 8 performance measures. Therefore the performances for all three algorithms are not statistically similar.

The bar charts in Figure 4.12 to Figure 4.18 in section 4.2 of the chapter indicate that the performance of the three algorithms had a worsened, relative to anomaly performance detection. The MLP and NaiveBayes performance was reduced from being an excellent 90% accuracy to a good to mediocre 70-80% accuracy.

The data-mining capabilities of the MLP and NaiveBayes may have not been statistically the same as shown by the performance measures in section 4.2 of the chapter, but were slightly similar as shown by the bar charts in Figure 4.22 and Figure 4.24.

The data-mining capabilities of the AIRS2 algorithms are revealed by the bar chart in figure 4.24, and are very different from those shown by the MLP and NaiveBayes in bar charts for Figure 4.22 and Figure 4.23. This can be reflective of the AIRS2's poor performance measures shown in section 4.2.

Overall, the MLP and NaiveBayes had a good aberrant performance measures, but not as excellent as those of the anomaly performance detection.

5. REDBUTTON NETWORK INTRUSION DETECTION CASE STUDY

5.1. Introduction

Case study 1 of the dissertation has revealed that there is an overwhelming growth in the use of Wi-Fi networks in airports, hotels and enterprises. It has also indicated the need for intelligent computational techniques to handle the massive monitoring data that is needed for the network monitoring. However, there is a greater challenge of avoiding network attacks. There have been prevention techniques such as firewalls, access control or encryption, but these have failed to fully protect networks from attacks [92]. Authors in [12] indicate that there are two main types of attacks to Wi-Fi networks, unintentional and intentional.

Unintentional attacks come from devices that use the same ISM frequency band using the 2.4GHz spectrum. These include devices like cell phones, GPS, RFID tags and Bluetooth devices. Other devices that are not used for communication such as microwave ovens transmit RF in this spectrum [12]. This type of transmission can disrupt Wi-Fi communication through interference and thus prevent the user from connecting to the Wi-Fi spot. With Intentional attacks a malicious user disrupts Wi-Fi communications through the use of a normal PC and software. This can cause denial-of-service and prevent access to the network [12].

In this case study we explore the intentional attacks where a hacker or attacker is sitting at a computer and sending erroneous data to a monitoring system. In case study 1, we looked at the performance of three machine learning algorithms, AIS, NaiveBayes and MLP on detecting anomalous and aberrant performance in a network. Again, we would like to explore the impact of an intentional attacker on a network, and the ability of the three machine learning algorithms to be able to detect anomaly and aberrant performance in the network.

5.2. Background and Related Work

According to [92], "an intrusion detection system dynamically monitors the events taking place in a system, and decides whether these events are systematic of an attack or constitute a legitimate use of the system". Intrusion detection systems use computational intelligence to classify what belongs as normal behaviour, and what is abnormal behaviour.

There has been much previous work in the field of AIS techniques applied to intrusion detection, and there is on-going research in this field. This work includes that of [89], where they examined the use of AIS to detect computer network intrusion. They found out that the AIS is able to discriminate from what belongs as normal behaviour and what does not; and the AIS was able to detect unknown attacks to the network. In [54], an AIS framework was

deployed on a larger peer-to-peer network. They found out that the AIS framework had a good detection rate and low false positive rate.

Similar work was produced using artificial neural networks, in [31], where they used an artificial neural network to learn the behaviour of a program in order to detect intrusive behaviour by a computer virus. In [38], neural networks were used to detect anomaly intrusion in TCP and IP header information for the data from the Defence Advanced Research Projects Agency (DARPA), and the neural network managed to achieve a best overall classification of 99.42%.

In [49], they used a Bayesian classification method to correctly classify network intrusions using the MIT Lincoln Labs 1999 datasets. Using Bayesian classification resulted in lower false positive rate performance when compared to traditional intrusion detection systems. The work done in [84, 85] was an extension of an open source intrusion detection system with Bayesian networks capabilities. 'Snort' is an open source network-based intrusion detection system that is able to perform protocol analysis and real-time traffic analysis and packet logging on internet IP networks. In this work it was combined with the use of open source C++ Bayesian networks systems (SMILE and GeNIe). The aim was to produce a lower false positive rate for the intrusion detection system.

Based on the success of the AIS, Bayesian Network and Artificial Neural Networks machine learning algorithm, and following the model used in the previous case study, we would like to implement the algorithms for monitoring a large-scale W-Fi hotspots network where the dataset used previously was altered to mimic intrusion. The performance of the three algorithms (AIRS2, NaiveBayes and MLP) is evaluated and statistically compared. In this evaluation and comparison we would like to answer the following research questions:

- Which method performs better?
- How do they perform under different test cases and network thresholds?
- How do the methods perform for anomaly detection, and aberrant behaviour detection?
- What kind of information can be mined using these machine learning algorithms?

5.3. Research Methods

In this case study, the RedButton network was used in order to gauge the impact of intentional intrusion on a Wi-Fi network, and the ability of the three machine learning algorithms to detect anomaly and aberrant behaviour after an intrusion in the network. In this work, the same data collection method, network performance metrics, detector encoding schemes where used. The performances of the three machine learning algorithms were tested against four test cases that were seen in Table 4.1 of case study 1.

The present case study uses an approach similar the RedButton case but following the model suggested by the work in [92] where it is suggested researchers in the field of intrusion

detection usually generate their own datasets by self-producing intrusion data and merging it with training datasets. Building upon this approach, when self-producing the data we mimicked a scenario where a malicious attacker would like to send extreme erroneous performance data to the syslog daemon tool, in this case, simulating a Denial-of-Service (DoS) attack to the tool. The self-produced data was then merged with the existing monitoring data, in such a way that two-thirds of the data were from the live network, while a third was from a self-produced malicious attacker, with extremely anomalous data.

In generating the anomalous data, an extreme level of anomaly was considered. The following Java pseudo code was used:

$$Noise = -30dB - (Math.random() * 50dB)$$

$$Load = 80\% + (Math.random() * 100\%)$$

$$Uptime = (Math.Random() * 60)$$

Using this pseudo code, the generated values of noise will fall between -30.00dB and -80.00db. This range of noise values are considered to be undesirable in a Wi-Fi network [26]. The load values will range between load levels of 80% to 180% when using the pseudo code above. Uptime will have a value below 60 minutes, indicating that the gateway device has recovered from a restart state.

5.4. Experiment Results

The results that follow in this section are from a study conducted to realise the effects of intrusion detection on a Wi-Fi monitoring network, and the ability of the MLP, NaiveBayes and AIRS2 algorithms in detecting anomaly and aberrant performance for an intruded network.

5.4.1. Anomaly Performance

In this study we investigate whether there is a significant difference in performance for the MLP, NaiveBayes and AIRS2 algorithms when detecting anomaly in an intruded network. An experiment was conducted and the following results are revealed using bar charts description, and hypothesis testing statistics. The study discusses the results based on the following performance measures: Kappa Statistic, True Positive Rate, False Positive Rate, Precision, Recall, F-measure, ROC Area, and data mined by the three algorithms.

5.4.1.1. Anomaly Kappa Statistic Performance

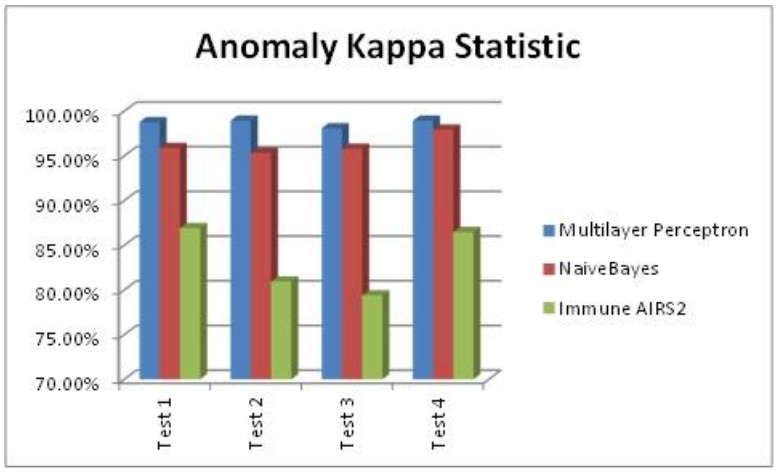

Figure 5.1: Anomaly Kappa Statistic Performance Bar Chart

Across all test cases, the MLP and NaiveBayes had an average anomaly Kappa statistic performance of 98.68% and 96.23%, respectively, while the AIRS2 had an average anomaly Kappa statistic of 83.44%. This is shown by the bar chart in Figure 5.1. The MLP Kappa statistic performance has been slightly above 98.00% across all test cases. The NaiveBayes started with a 95.88% in test case 1, this kept improving until 97.93% in test case 4. The AIRS2 had a good Kappa Statistic performance; the lowest performance was 79.40% in test case 3 and the highest was 86.92% in test case 1.

Table 5.1: Results for Anomaly Kappa Statistic T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.961525529	0.962428791
Variance	0.000299294	0.000130784
Observations	200	200
Pearson Correlation	0.564472653	
Hypothesized Mean Difference	0	
df	199	
t Stat	-0.888456961	
P(T<=t) one-tail	0.187683817	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	0.375367635	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.961525529	0.814756869
Variance	0.000299294	0.004972706
Observations	200	200
Pearson Correlation	0.366653	
Hypothesized Mean Difference	0	
df	199	
t Stat	31.3718675	
P(T<=t) one-tail	2.86616E-79	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	5.73232E-79	
t Critical two-tail	1.971956544	

For the test:

Anomaly Kappa Statistic: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly Kappa Statistic: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -0.888 and its two-tailed p-value is 0.375. At 5% confidence level, the test is not statistically significant and there is little to no evidence to infer that the alternative hypothesis is true. Therefore we do not reject the null hypothesis and conclude that the difference in the mean anomaly Kappa Statistic for the MLP and NaiveBayes algorithms is equal to zero, the hypothesised mean.

For the test:

Anomaly Kappa Statistic: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly Kappa Statistic: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 31.371 and its two-tailed p-value is 5.732E-179. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly Kappa Statistic for the MLP and AIRS2 algorithms.

5.4.1.2. Anomaly True Positive Rate Performance

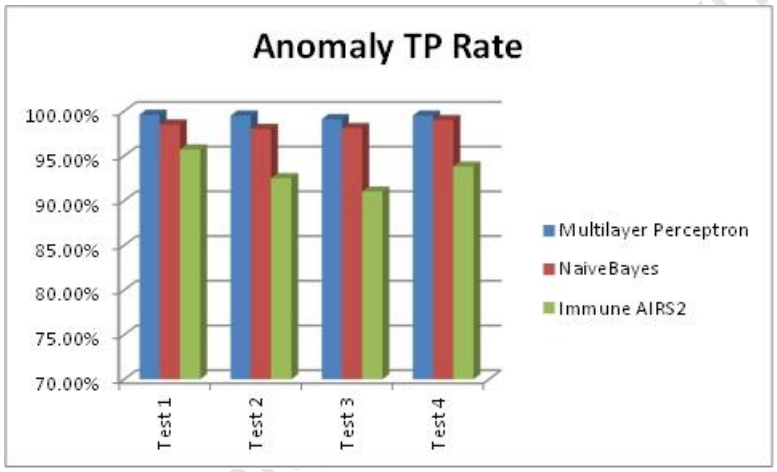

Figure 5.2: Anomaly True Positive Rate Performance Bar Chart

The MLP and the NaiveBayes had an anomaly true positive rate of 99.43% and 98.40%, respectively, while the AIRS2 had an average anomaly true positive rate of 93.25% across all test cases. The TP rate kept declining from test cases 1 to 3 and a sudden peak in test case 4. This is revealed by Figure 5.2 above. The anomaly TP rate performance for the MLP and NaiveBayes was very good, above 98% for all test cases. The AIRS2 also had a good performance, with the lowest being 91.00% in test case 3 and the highest being 95.70% in test case 1.

Table 5.2: Results for Anomaly True Positive Rate T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.98369182	0.983879568
Variance	5.00919E-05	2.1228E-05
Observations	200	200
Pearson Correlation	0.56040514	
Hypothesized Mean Difference	0	
df	199	
t Stat	-0.450275672	
P(T<=t) one-tail	0.326500914	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	0.653001829	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.98369182	0.926483647
Variance	5.00919E-05	0.000743581
Observations	200	200
Pearson Correlation	0.379600485	
Hypothesized Mean Difference	0	
df	199	
t Stat	31.80318183	
P(T<=t) one-tail	2.96795E-80	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	5.9359E-80	
t Critical two-tail	1.971956544	

For the test:

Anomaly TP Rate: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly TP Rate: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -0.450 and its two-tailed p-value is 0.653. At 5% confidence level, the test is not statistically significant and there is little or no evidence to infer that the alternative hypothesis is true. Therefore we do not reject the null hypothesis and conclude that the difference in the mean anomaly True Positive Rate between the MLP and NaiveBayes algorithms equals zero, the hypothesised mean difference.

For the test:

Anomaly TP Rate: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly TP Rate: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 31.803 and its two-tailed p-value is 5.9359E-80. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly True Positive Rate for the MLP and AIRS2 algorithms.

5.4.1.3. Anomaly False Positive Performance

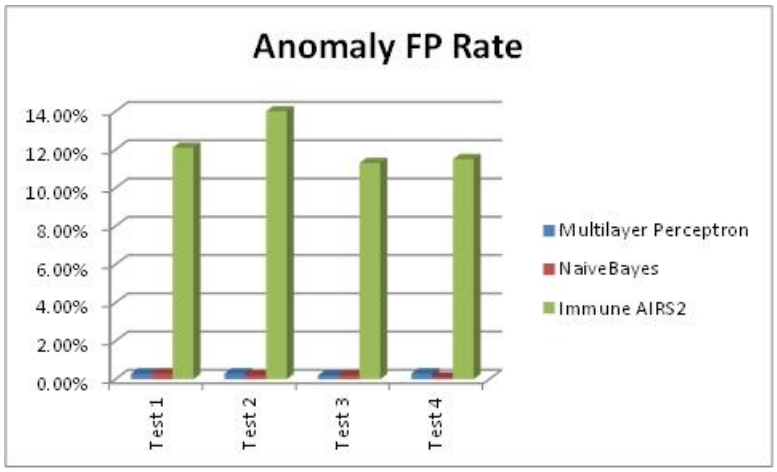

Figure 5.3: Anomaly False Positive Rate Performance Bar Chart

Figure 5.3 above reveals that across all test cases, the MLP and NaiveBayes had very low anomaly false positive rate of 0.28% and 0.20%, while the average anomaly false positive rate of the AIRS2 algorithm was 12.23%. The anomaly FP rate performance for the AIRS2 was always above 10.0%, for all test cases.

Table 5.3: Results for Anomaly False Positive Rate T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.007876793	0.002010208
Variance	9.79882E-05	8.52944E-07
Observations	200	200
Pearson Correlation	0.168305493	
Hypothesized Mean Difference	0	
df	199	
t Stat	8.478122494	
P(T<=t) one-tail	2.55882E-15	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	5.11764E-15	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.007876793	0.13484822
Variance	9.79882E-05	0.004879868
Observations	200	200
Pearson Correlation	0.054631526	
Hypothesized Mean Difference	0	
df	199	
t Stat	-25.64608033	
P(T<=t) one-tail	2.667E-65	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	5.334E-65	
t Critical two-tail	1.971956544	

For the test:

Anomaly FP Rate: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly FP Rate: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 8.478 and its two-tailed p-value is 5.11764E-15. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly False Positive Rate for the MLP and NaiveBayes algorithms.

For the test:

Anomaly FP Rate: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly FP Rate: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is -25.646 and its two-tailed p-value is 5.334E-65. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly False Positive Rate for the MLP and AIRS2 algorithms.

5.4.1.4. Anomaly Precision Performance

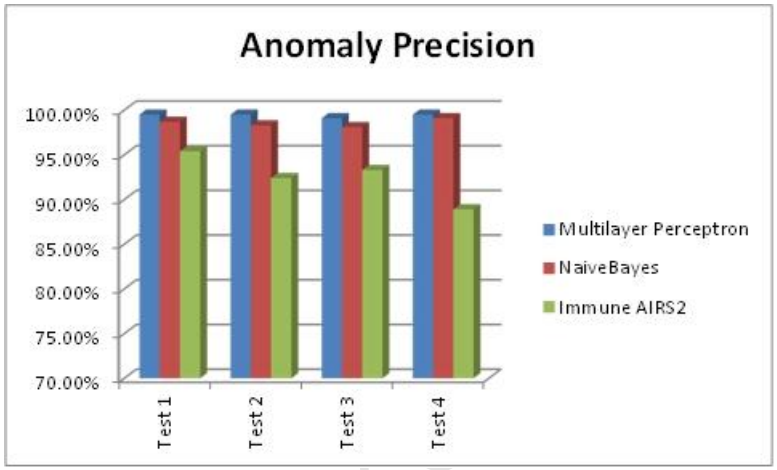

Figure 5.4: Anomaly Precision Performance Bar Chart

The MLP and NaiveBayes had an average anomaly precision performance of 99.40% and 98.55%, respectively, while the AIRS2 had an average anomaly performance of 92.50% across all test cases. Figure 5.4 reveals that the algorithms' precision values declined in test cases 1, 2 and 3, while there was an increase in test case 3. The anomaly Precision for both MLP and NaiveBayes was above 98.0% for all test cases. The AIRS2 had its low performance at 88.90% in test case 4 and the highest performance in test case 95.40%.

Table 5.4: Results for Anomaly Precision T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.981034903	0.985442166
Variance	6.05762E-05	1.94819E-05
Observations	200	200
Pearson Correlation	0.713248343	
Hypothesized Mean Difference	0	
df	199	
t Stat	-11.18482464	
P(T<=t) one-tail	3.78054E-23	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	7.56107E-23	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.981034903	0.891872234
Variance	6.05762E-05	0.001604549
Observations	200	200
Pearson Correlation	0.250190364	
Hypothesized Mean Difference	0	
df	199	
t Stat	32.45903269	
P(T<=t) one-tail	9.81821E-82	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.96364E-81	
t Critical two-tail	1.971956544	

For the test:

Anomaly Precision: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly Precision: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -11.184 and the two-tailed p-value is 7.56107E-23. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly precision for the MLP and NaiveBayes algorithms.

For the test:

Anomaly Precision: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly Precision: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 32.459 and the two-tailed p-value is 1.96364E-81. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is difference in the mean anomaly precision for the MLP and AIRS2 algorithms.

5.4.1.5. Anomaly Recall Performance

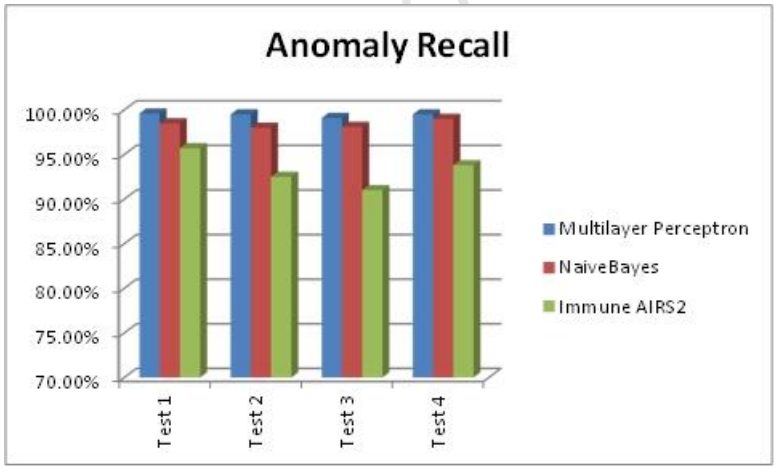

Figure 5.5: Anomaly Recall Performance Bar Chart

Across all test cases, the average anomaly precision performance for the MLP and NaiveBayes was 99.43% and 98.40%, respectively, while the AIRS2 had an average anomaly precision of 93.25%. This is revealed by the Figure 5.5 above. The MLP and NaiveBayes had a very good performance above 95% across all test cases. The AIRS2 had 91.00% performance in test case 3 and its highest was 95.70% in test case 1.

Table 5.5: Results for Anomaly Recall T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.98369182	0.983879568
Variance	5.00919E-05	2.1228E-05
Observations	200	200
Pearson Correlation	0.56040514	
Hypothesized Mean Difference	0	
df	199	
t Stat	-0.450275672	
P(T<=t) one-tail	0.326500914	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	0.653001829	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.98369182	0.926483647
Variance	5.00919E-05	0.000743581
Observations	200	200
Pearson Correlation	0.379600485	
Hypothesized Mean Difference	0	
df	199	
t Stat	31.80318183	
P(T<=t) one-tail	2.96795E-80	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	5.9359E-80	
t Critical two-tail	1.971956544	

For the test:

Anomaly Recall: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly Recall: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -0.450 and its two-tailed p-value is 0.653. At 5% confidence level, the test is not statistically significant and there is little or no evidence to infer that the alternative hypothesis is true. Therefore we do not reject the null hypothesis and conclude that the difference in the mean anomaly recall for the MLP and NaiveBayes algorithms equals zero, the hypothesised mean.

For the test:

Anomaly Recall: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly Recall: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 31.803 and its two-tailed p-value is 5.9395E-80. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly recall for the MLP and AIRS2 algorithms.

5.4.1.6. Anomaly F-measure Performance

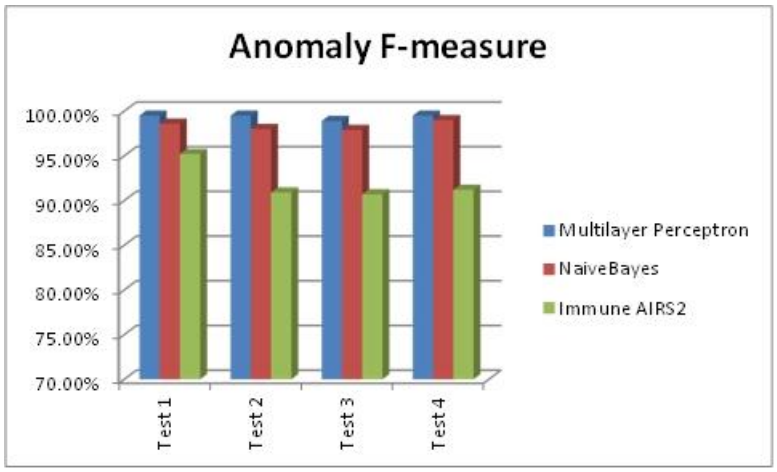

Figure 5.6: Anomaly F-measure Performance Bar Chart

For anomaly F-measure performance, The MLP and NaiveBayes had an average performance of 99.35% and 98.38%, respectively, while the AIRS2 had an average anomaly f-measure of 92.00% across all test cases. This is revealed by the bar chart in Figure 5.6. The overall anomaly F-measure performance for all algorithms was very good; the MLP and the NaiveBayes had performance above 98 % for all test cases. The AIRS2 had its high f-measure performance of 95.20% in test case 1; this kept declining until it got to an f-measure performance of 91.20% in test case 4.

Table 5.6: Results for Anomaly F-Measure T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.981277635	0.983666774
Variance	6.01052E-05	2.47133E-05
Observations	200	200
Pearson Correlation	0.604074395	
Hypothesized Mean Difference	0	
df	199	
t Stat	-5.462733901	
P(T<=t) one-tail	6.95168E-08	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.39034E-07	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.981277635	0.904885382
Variance	6.01052E-05	0.001149517
Observations	200	200
Pearson Correlation	0.420350302	
Hypothesized Mean Difference	0	
df	199	
t Stat	34.35935525	
P(T<=t) one-tail	6.54526E-86	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.30905E-85	
t Critical two-tail	1.971956544	

For the test:

Anomaly F-Measure: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly F-Measure: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -5.462 and its two-tailed p-value is 1.390434E-07. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly F-measure for the MLP and NaiveBayes algorithms.

For the test:

Anomaly F-Measure: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly F-Measure: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 34.359 and its two-tailed p-value is 1.30905E-85. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly F-measure for the MLP and AIRS2 algorithms.

5.4.1.7. Anomaly ROC Area Performance

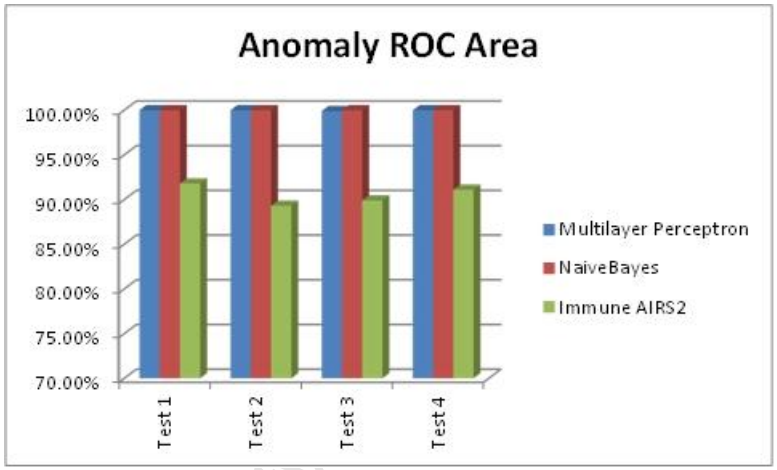

Figure 5.7: Anomaly ROC Area Performance Bar Chart

The bar chart in Figure 5.7 reveals that across all test cases the average anomaly ROC Area performance for the MLP and NaiveBayes was 99.98% and 100.00%, respectively, while that of the AIRS2 was slightly lower at 90.53%. The NaiveBayes had a very good ROC Area performance achieving 100% for all test cases; the MLP had a similar performance only for test case 3 it achieved a 99.90% ROC area performance. The AIRS2 had its lowest ROC area performance of 89.30% in test case 2 and it highest performance of 91.80% in test case 1. Overall the algorithms had a good ROC area performance.

Table 5.7: Results for Anomaly ROC Area T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.999852319	0.999644457
Variance	1.82481E-08	1.06718E-07
Observations	200	200
Pearson Correlation	0.389777899	
Hypothesized Mean Difference	0	
df	199	
t Stat	9.76813576	
P(T<=t) one-tail	5.82589E-19	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.16518E-18	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.999852319	0.895817714
Variance	1.82481E-08	0.001951289
Observations	200	200
Pearson Correlation	0.135881352	
Hypothesized Mean Difference	0	
df	199	
t Stat	33.32042758	
P(T<=t) one-tail	1.198E-83	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	2.396E-83	
t Critical two-tail	1.971956544	

For the test:

Anomaly ROC Area: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Anomaly ROC Area: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 9.768 and its two-tailed p-value is 1.16518E-18. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly ROC Area for the MLP and NaiveBayes algorithms.

For the test:

Anomaly ROC Area: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Anomaly ROC Area: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 33.320 and its two-tailed p-value is 2.396E-83. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean anomaly ROC Area for the MLP and AIRS2 algorithms.

5.4.2. Aberrant Performance

Another experiment conducted in this study was to investigate whether there is a significant difference in the ability of the MLP, NaiveBayes and AIRS2 algorithms to detect aberrant behaviour in an intruded network. To do this, we present the experiment results using bar charts descriptions, and hypothesis testing statistics. The study discusses the results based on the following performance measures: Kappa Statistic, True Positive Rate, False Positive Rate, Precision, Recall, F-measure, ROC Area, and data mined by the three algorithms.

5.4.2.1. Aberrant Kappa Statistic Performance

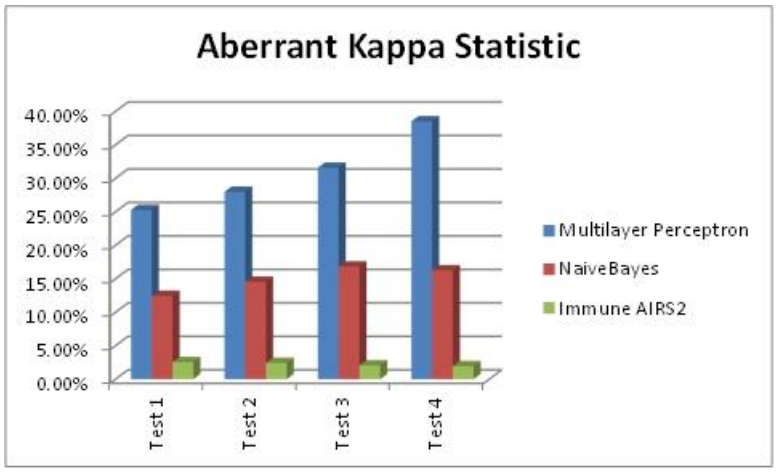

Figure 5.8: Aberrant Kappa Statistic Performance Bar Chart

Across all test cases, the average aberrant Kappa statistic for the MLP and NaiveBayes algorithms was low at 30.85% and 15.03%, respectively, while that of the AIRS2 algorithm was even lower at 2.25%. This is show by the bar chart in figure 5.8. Kappa statistic performance for the MLP started at 25.29% in test case 1, and this kept improving to a 38.51% in test case 4. A similar trend in performance was displayed by the NaiveBayes, in test case 1 its Kappa Statistic performance was 12.44% and this kept improving until a 16.26% in test case 4. The overall Kappa Statistic performance for the three algorithms was poor.

Table 5.8: Results for Aberrant Kappa Statistic T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.019777764	0.143432174
Variance	0.004467718	0.000723831
Observations	200	200
Pearson Correlation	-0.076185274	
Hypothesized Mean Difference	0	
df	199	
t Stat	-23.65415314	
P(T<=t) one-tail	4.95985E-60	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	9.91971E-60	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.019777764	0.021326453
Variance	0.004467718	2.59674E-05
Observations	200	200
Pearson Correlation	0.286114208	
Hypothesized Mean Difference	0	
df	199	
t Stat	-0.334046249	
P(T<=t) one-tail	0.36934821	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	0.738696419	
t Critical two-tail	1.971956544	

For the test:

Aberrant Kappa Statistic: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant Kappa Statistic: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -23.654 and its two-tailed p-value is 9.91971E-60. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude

that there is a difference in the mean aberrant Kappa statistic for the MLP and NaiveBayes algorithms.

For the test:

Aberrant Kappa Statistic: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant Kappa Statistic: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is -0.334 and its two-tailed p-value is 0.739. At 5% confidence level, the test is not statistically significant and there is little or no evidence to infer that the alternative hypothesis is true. Therefore we do not reject the null hypothesis and conclude that the difference in the mean aberrant Kappa statistic for the MLP and AIRS2 algorithms is equal to zero, the hypothesised mean.

5.4.2.2. Aberrant True Positive Rate Performance

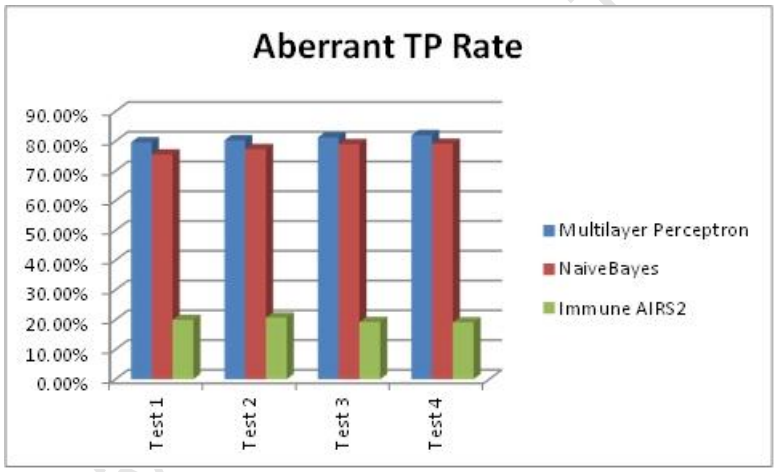

Figure 5.9: Aberrant TP Rate Performance Bar Chart

The MLP and the NaiveBayes had an average aberrant TP Rate performance of 80.80% and 77.73%, respectively, while the NaiveBayes had an average true positive rate performance of 19.73% across all test cases. The aberrant TP rate for the MLP started with a 75.50% performance in test case 1, and this kept improving until an 82.00% performance in test case 4. The performance of the NaiveBayes had a similar trend in performance, it was 75.50% in test case 1 and it kept improving to 79.10% performance in test case 4. The AIRS2 had a reverse trend in performance, from test case 1 to 4, the performance kept declining and it was below 20% for all test cases.

Table 5.9: Results for Aberrant True Positive Rate T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.780733858	0.776527334
Variance	4.8016E-05	0.000224476
Observations	200	200
Pearson Correlation	0.13357897	
Hypothesized Mean Difference	0	
df	199	
t Stat	3.802523783	
P(T<=t) one-tail	9.52165E-05	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	0.000190433	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.780733858	0.195577384
Variance	4.8016E-05	0.000205134
Observations	200	200
Pearson Correlation	0.225194417	
Hypothesized Mean Difference	0	
df	199	
t Stat	573.1726842	
P(T<=t) one-tail	0	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	0	
t Critical two-tail	1.971956544	

For the test:

Aberrant True Positive Rate: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant True Positive Rate: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 3.802 and its two-tailed p-value is 0.0001. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant true positive rate for the MLP and NaiveBayes algorithms.

For the test:

Aberrant True Positive Rate: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant True Positive Rate: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 573.172 and its two-tailed p-value is 0. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant true positive rate for the MLP and AIRS2 algorithms.

5.4.2.3. Aberrant False Positive Rate Performance

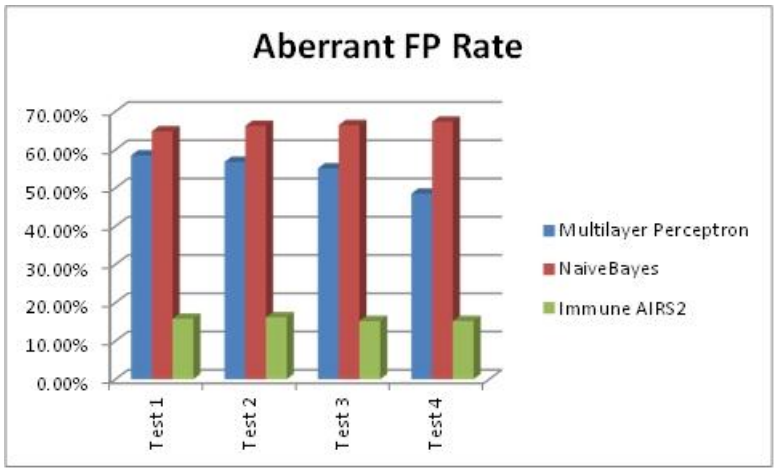

Figure 5.10: Aberrant False Positive Rate Performance Bar Chart

The bar chart in figure 5.10 reveals that the MLP and NaiveBayes had a high average aberrant false positive rate performance of 54.73% and 66.18%, respectively. The aberrant FP rate performance for both MLP and NaiveBayes was high compared to that of the AIRS2. AIRS2's aberrant FP rate performance was low at an average of 15.60% across all test cases.

Table 5.10: Results for Aberrant False Positive Rate T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.765061869	0.666413075
Variance	0.002291394	0.000517095
Observations	200	200
Pearson Correlation	0.311684921	
Hypothesized Mean Difference	0	
df	199	
t Stat	30.22894601	
P(T<=t) one-tail	1.29137E-76	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	2.58274E-76	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.765061869	0.156820725
Variance	0.002291394	0.000157068
Observations	200	200
Pearson Correlation	-0.026133509	
Hypothesized Mean Difference	0	
df	199	
t Stat	172.7352054	
P(T<=t) one-tail	4.5942E-219	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	9.1884E-219	
t Critical two-tail	1.971956544	

For the test:

Aberrant False Positive Rate: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant False Positive Rate: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 30.228 and its two-tailed p-value is 2.58274E-76. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant false positive rate for the MLP and NaiveBayes algorithms.

For the test:

Aberrant False Positive Rate: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant False Positive Rate: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 172.735 and its two-tailed p-value is 9.1884E-219. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant false positive rate for the MLP and AIRS2 algorithms.

5.4.2.4. Aberrant Precision Performance

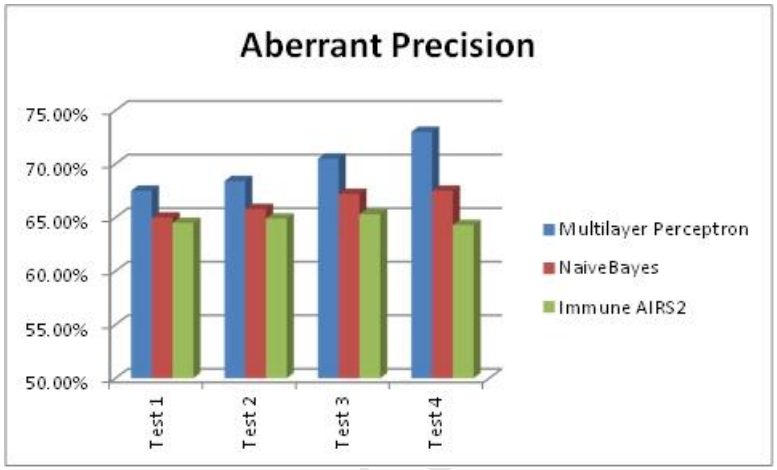

Figure 5.11: Aberrant Precision Bar Chart

Figure 5.11 above reveals that across all test cases, the MLP and NaiveBayes algorithms had an average aberrant precision performance of 69.85% and 66.38%, respectively, while that of the AIRS2 algorithms was 64.75%. The MLP has a better aberrant precision performance of 73.00% in test case 4. For all algorithms had average aberrant precision performance ranging between 65%-70%.

Table 5.11: Results for Aberrant Precision T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.6112358	0.657257327
Variance	0.000238185	0.000117589
Observations	200	200
Pearson Correlation	0.101236847	
Hypothesized Mean Difference	0	
df	199	
t Stat	-36.27631909	
P(T<=t) one-tail	5.84565E-90	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.16913E-89	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.6112358	0.64609771
Variance	0.000238185	0.000142404
Observations	200	200
Pearson Correlation	0.523894841	
Hypothesized Mean Difference	0	
df	199	
t Stat	-35.99391098	
P(T<=t) one-tail	2.25626E-89	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	4.51253E-89	
t Critical two-tail	1.971956544	

For the test:

Aberrant Precision: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant Precision: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -36.276 and its two-tailed p-value is 1.16913E-89. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant precision for the MLP and NaiveBayes algorithms.

For the test:

Aberrant Precision: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant Precision: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is -35.993 and its two-tailed p-value is 1.16913E-89. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant precision for the MLP and AIRS2 algorithms.

5.4.2.5. Aberrant Recall Performance

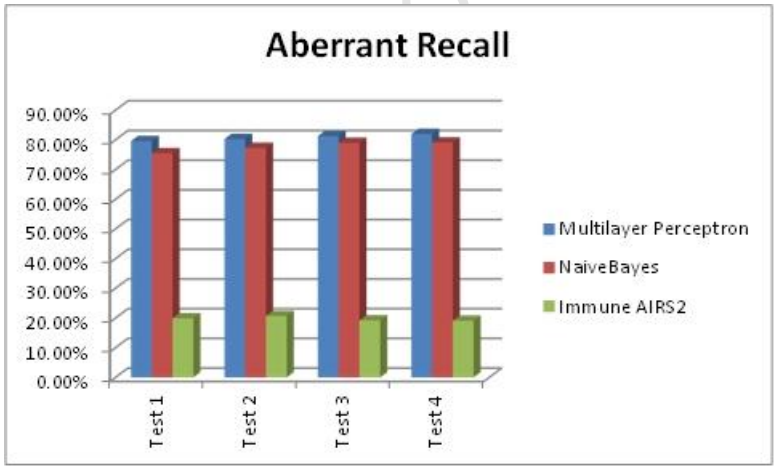

Figure 5.12: Aberrant Recall Performance Bar Chart

The MLP and NaiveBayes algorithms had an average aberrant recall performance of 80.80% and 77.73%, respectively, while that of the NaiveBayes was low at 19.73% across all test cases, as shown by Figure 5.12 above.

Table 5.12: Results for Aberrant Recall T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.780733858	0.776527334
Variance	4.8016E-05	0.000224476
Observations	200	200
Pearson Correlation	0.13357897	
Hypothesized Mean Difference	0	
df	199	
t Stat	3.802523783	
P(T<=t) one-tail	9.52165E-05	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	0.000190433	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.780733858	0.195577384
Variance	4.8016E-05	0.000205134
Observations	200	200
Pearson Correlation	0.225194417	
Hypothesized Mean Difference	0	
df	199	
t Stat	573.1726842	
P(T<=t) one-tail	0	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	0	
t Critical two-tail	1.971956544	

For the test:

Aberrant Recall: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant Recall: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 3.802 and its two-tailed p-value is 0.0001. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant recall for the MLP and NaiveBayes algorithms.

For the test:

Aberrant Recall: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant Recall: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 573.172 and its two-tailed p-value is 0. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant recall for the MLP and AIRS2 algorithms.

5.4.2.6. Aberrant F-measure Performance

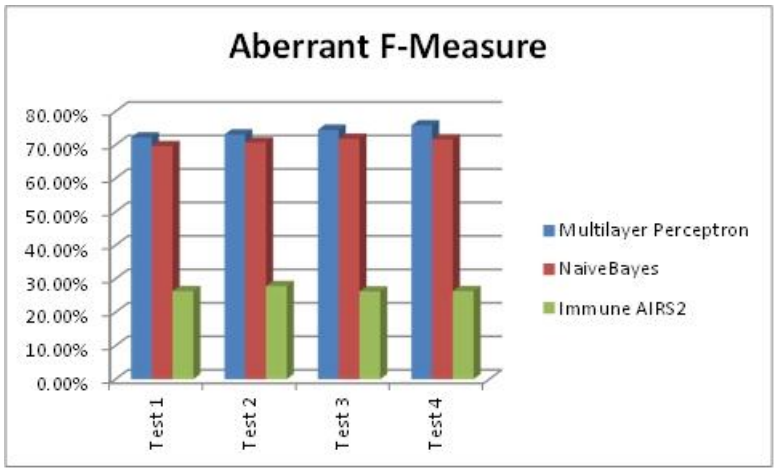

Figure 5.13: Aberrant F-measure Performance Bar Chart

The bar chart in Figure 5.13 above reveals that the MLP and NaiveBayes algorithms had an average aberrant f-measure of 73.90% and 70.93%, respectively, while that of the AIRS2 algorithm was lower at 26.65%.

Table 5.13: Results for Aberrant F-measure T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.685491515	0.70705448
Variance	0.000139686	8.73708E-05
Observations	200	200
Pearson Correlation	0.039904466	
Hypothesized Mean Difference	0	
df	199	
t Stat	-20.64219598	
P(T<=t) one-tail	1.18043E-51	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	2.36087E-51	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.685491515	0.264267914
Variance	0.000139686	0.000334531
Observations	200	200
Pearson Correlation	0.294113218	
Hypothesized Mean Difference	0	
df	199	
t Stat	319.7604902	
P(T<=t) one-tail	4.3962E-272	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	8.7925E-272	
t Critical two-tail	1.971956544	

For the test:

Aberrant F-measure: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant F-measure: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -20.642 and its two-tailed p-value is 2.36087E-51. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant F-measure for the MLP and NaiveBayes algorithms.

For the test:

Aberrant F-measure: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant F-measure: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 319.760 and its two-tailed p-value is 8.7925E-272. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant F-measure for the MLP and AIRS2 algorithms.

5.4.2.7. Aberrant ROC Area Performance

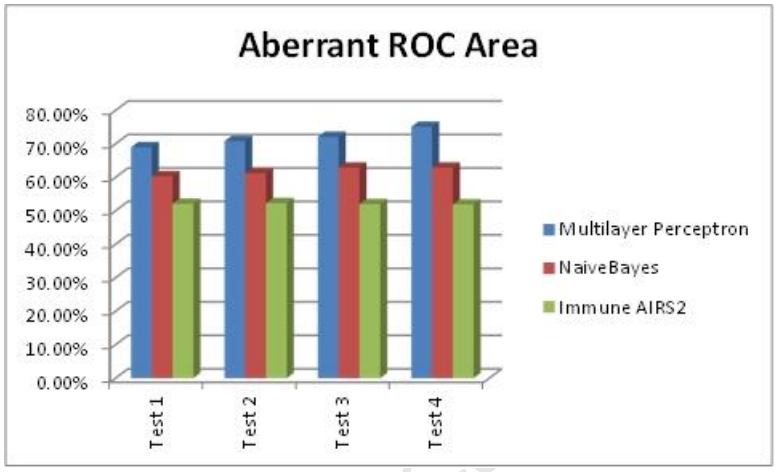

Figure 5.14: Aberrant ROC Area Performance Bar Chart

Across all test cases, the MLP and NaiveBayes had and average aberrant ROC area performance of 71.75% and 61.83%, respectively, while the AIRS2 algorithm had an average aberrant ROC area performance of 52.08%. This is shown by the bar chart in Figure 5.14. The MLP had a 69.00% ROC area performance in test case 1 and this kept improving to a 75.10% in test case 4. A similar trend was seen with NaiveBayes where is had a 60.30% performance in test case 1 and it kept improving to a slightly higher 62.90% performance in test case 4.

Table 5.14: Results for Aberrant ROC Area T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.567574139	0.611586285
Variance	0.002629827	0.001341779
Observations	200	200
Pearson Correlation	0.937465214	
Hypothesized Mean Difference	0	
df	199	
t Stat	-29.35425077	
P(T<=t) one-tail	1.53504E-74	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	3.07008E-74	
t Critical two-tail	1.971956544	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.567574139	0.519378329
Variance	0.002629827	3.41121E-05
Observations	200	200
Pearson Correlation	0.311801688	
Hypothesized Mean Difference	0	
df	199	
t Stat	13.69453144	
P(T<=t) one-tail	8.2364E-31	
t Critical one-tail	1.652546746	
P(T<=t) two-tail	1.64728E-30	
t Critical two-tail	1.971956544	

For the test:

Aberrant ROC Area: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Aberrant ROC Area: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -29.354 and its two-tailed p-value is 3.07008E-74. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant ROC Area for the MLP and NaiveBayes algorithms.

For the test:

Aberrant ROC Area: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Aberrant ROC Area: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 13.694 and its two-tailed p-value is 1.64728E-30. At 5% confidence level, the test is highly significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean aberrant ROC Area for the MLP and AIRS2 algorithms.

5.4.3. Data Mining Using a Confusion Matrix

This section of the results chapter reveals data mining capabilities for the three algorithms. Using bar charts, we will discuss anomaly and aberrant behaviour detection in an intruded network.

5.4.3.1. Anomaly Performance Detection with Intrusion

This section discusses anomaly performance detection of the three algorithms in an intruded large Wi-Fi network.

5.4.3.1.1. Multilayer Perceptron Anomaly Detection

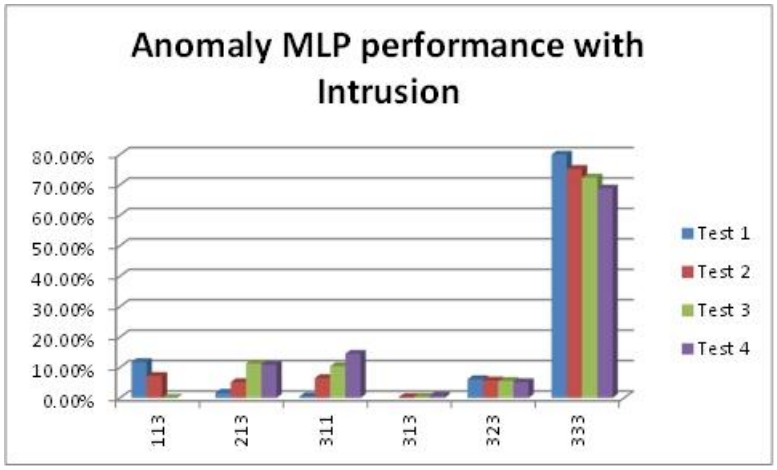

Figure 5.15: Multilayer Perceptron Anomaly Network Performance Detection Bar Chart

The MLP was able to identify 6 detectors in the intrusion network experiment. Among those identified detectors, the '333' had the highest detection rate. The '333' represent those devices that had experienced high and bad performance in noise, load, and downtime levels. The '333' had a detection rate between 79.98% in test case 1, and this kept decreasing to 68.78% in test case 4. This makes an average detection rate of 74.10% for the '333' detector.

The '113' represents those devices in the intruded network that had experienced high levels of downtime with low levels of noise and load performance. The detection rate for the '113' was 11.75% in test case 1 but this kept decreasing to 0.04% in test case 3, and 0% in test case 4. This makes an average detection rate of 6.32% across all test cases.

The '213' detector represents those devices that had experienced high levels of downtime with low load and moderate noise performance. The '213' had a detection rate of 1.67% in test case 1, and this kept increasing until a 10.92% in test case 4. This makes an average detection rate of 7.26% across all test cases.

The '311' represents those devices in the intruded network that had experienced high levels of noise, with low load and downtime performance. The '311' had a detection rate of 0.48% in test case 1, and this kept increasing to a 14.47% in test case 4, thus making an average detection rate of 7.96% across all intrusion detection test cases.

The '313' had the lowest average detection rate among those identified. It represents those devices in the network that had experienced high levels of noise and downtime, with low load performance. The '313' had a detection rate of 0% in test case 1, and this kept slightly increasing until a 0.66% in test case 4. This makes an average detection rate of 0.41% across all test cases.

The '323' represents those devices in the intruded network that had experienced high levels of both noise and downtime. The '323' had a detection rate of 6.12% in test case 1 and this kept slightly decreasing until 5.17% in test case 4. It had an average detection rate of 5.64%.

5.4.3.1.2. NaiveBayes Anomaly Detection

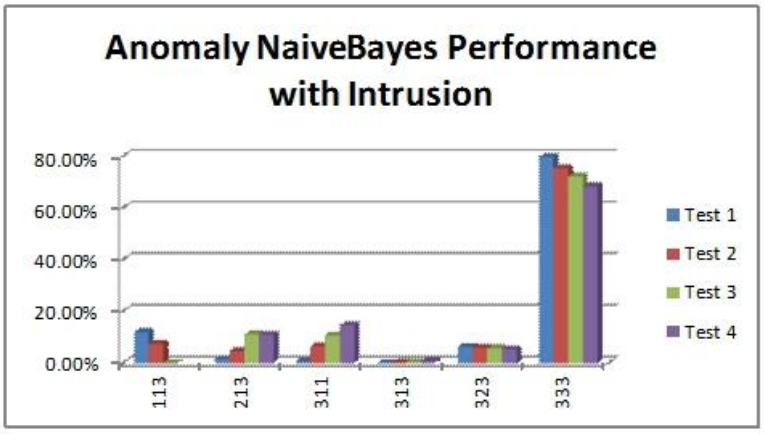

Figure 5.16: NaiveBayes Anomaly Network Performance Detection Bar Chart

The NaiveBayes identified similar detectors as those identified by the MLP. The only difference with the NaiveBayes was the detection rate of each detector.

The '333' had the highest average detection rate of 73.94% slightly lower than the detection rate identified by the MLP. The '113' and '213' had an average detection rate of 6.41% and 7.00%, respectively, while the '311' and '323' had an average detection rate of 8.11% and 5.78%, respectively. The '313' had the lowest average detection rate of 0.38% across all test cases.

5.4.3.1.3. AIRS2 Anomaly Detection

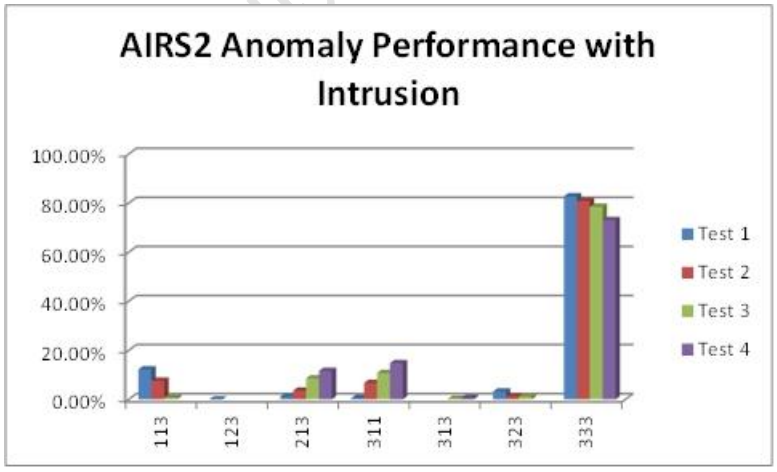

Figure 5.17: AIRS2 Anomaly Network Performance Detection Bar Chart

The AIRS2 was able to identify 6 detectors similar to those identified by the MLP and NaiveBayes algorithms. Again, the difference was in the detection rate of each detector from the three algorithms.

The '333' had an average detection rate of 78.79% this is higher than the detection rate identified by the NaiveBayes algorithm. The '113' and '313' had an average detection rate of

6.86% and 0.40%, respectively. The '213' detector was only identified on test case 1 of the experiments. The '311' and '323' had an average detection rate of 8.21% and 1.89%, respectively. The '213' had the lowest average detection rate of 0.00% across all test cases.

5.4.3.2. Aberrant Behaviour Detection with Intrusion

5.4.3.2.1. Multilayer Perceptron Aberrant Detection

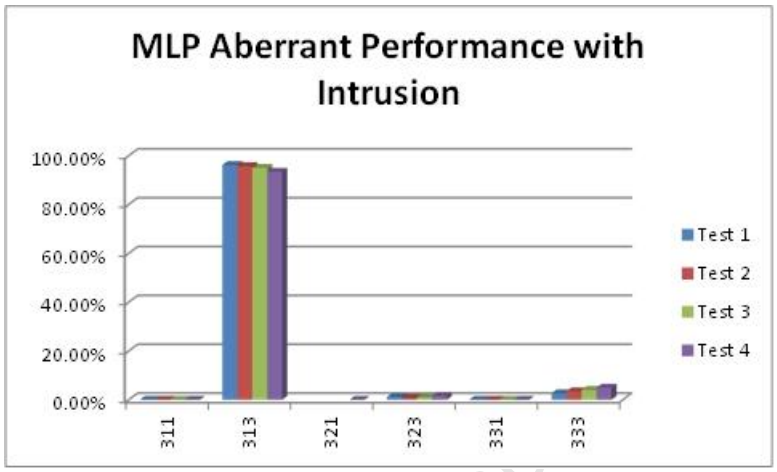

Figure 5.18: MLP Aberrant Network Behavior Detection Bar Chart

The MLP was able to identify 6 detectors for aberrant network behaviour when intrusion was introduced in the network. Among those identified the '313' had the highest detection rate. In test case 1, its detection rate was 96.17%, and this kept decreasing to a 93.40% in test case 4, thus having an average detection rate of 95.07% across all test cases.

The '323' had a detection rate of 1.13% in test case 1; this kept decreasing to a 0.90% in test case 4, making an average detection rate of 1.11% across all test cases. This was less than the average detection rate of the '333' of 3.78%. In test case 1 the '333' had a detection rate of 2.67% and as the networks threshold were made more stringent to test case 4, the detection rate increased to 4.94% in test case 4.

The '311', '321' and '331' had the lowest average detection rate of 0.02%, 0.00% and 0.02% respectively.

5.4.3.2.2. NaiveBayes Aberrant Detection

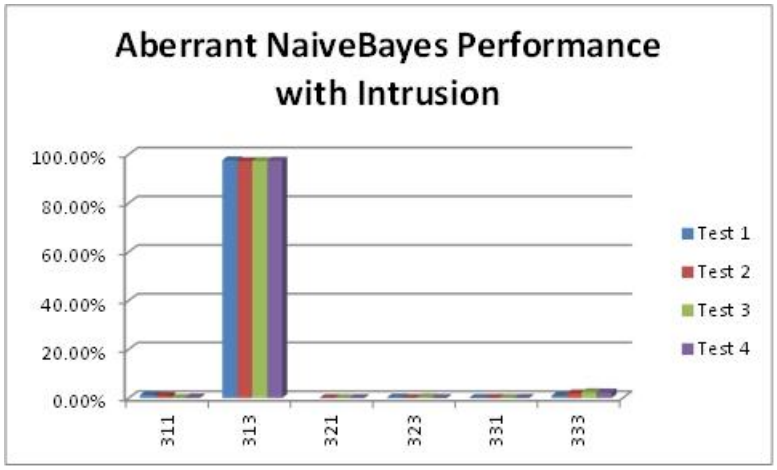

Figure 5.19: NaiveBayes Aberrant Network Behavior Detection Bar Chart

The NaiveBayes was able to identify 6 aberrant behaviour detectors, similar to those identified by the MLP algorithm. The only difference was with the detection rate of each detector. The '313' had an average detection rate of 97.38%, in test case 1 it had a detection rate of 97.59% and this kept decreasing to a 97.42% in test case 4.

The '333' had an average detection rate of 1.88%, in test case 1 the detection rate was 1.01% and this kept increasing to a 2.28% in test case 4. The '311' had an average detection rate of 0.58% across all test cases. In test case 1, its detection rate was 1.19% and this kept decreasing to a 0.24% in test case 4.

The '323', '331' and '321' had the lowest average detection rate for aberrant network behaviour of 0.10%, 0.05% and 0.00% respectively.

5.4.3.2.3. AIRS2 Aberrant Detection

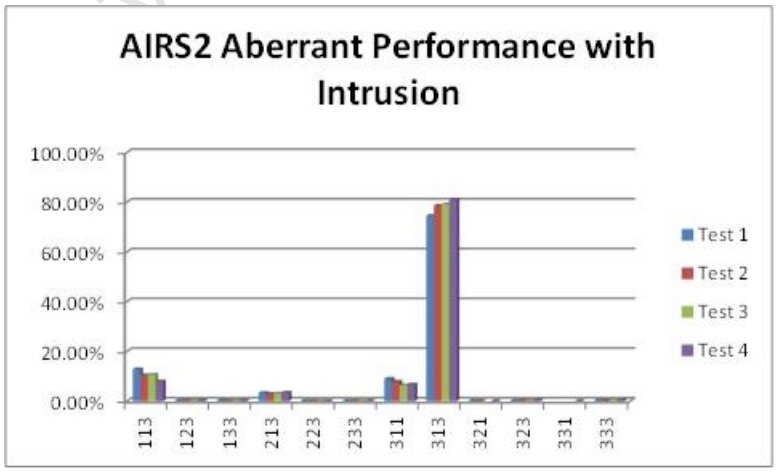

Figure 5.20: AIRS2 Aberrant Network Behavior Detection Bar Chart

The AIRS2 was able to identify 12 aberrant behaviour detectors in the intruded network. The detection rate of the '313' was lower than that of the NaiveBayes and MLP algorithms. The

‘313’ had an average detection rate of 78.27%, in test case 1 the detection rate was 74.46% and this kept increasing to an 81.04% in test case 4.

The second most identified detector was the ‘113’ with an average detection rate of 10.41% across all test cases. In test case 1, the detection rate was 12.80% and this kept decreasing to a 7.88% in test case 4. The ‘311’ followed with an average detection rate of 7.41%. In test case1 it had a detection rate of 8.93% and as the network performance threshold were made more stringent, the detection rate decreased to 6.63%.

In test case 1, the ‘213’ had a detection rate of 3.25%. This decreased to 2.84% in test case 2, and eventually increased to a 3.42% in test case 4.

The remaining detectors had an average detection rate less than 0.20% across all test cases. This includes detectors: ‘123’, ‘133’, ‘223’, ‘233’, ‘321’, ‘323’, ‘311’ and ‘333’. The ‘321’ was not identified in test case 3, and the ‘333’ was only identified in test 4 of the experiments.

5.5. Discussion

To test the MLP algorithm, Ghosh et al [30] ran experiments for anomaly detection and misuse detection, were they used the Lincoln DARPA intrusion detection datasets. In their anomaly detection results it was found that the MLP was able to perform with an average true positive rate of 77.3% and a false positive rate of 2.2%.

For misuse detection, the MLP was able to perform at 90.9% true positive rate and false positive rate of 18.7%. The author further explained the algorithm’s performance for misuse detection as due to the fact that the detection algorithm was trained on data that contained both intrusion data and normal data. Researchers have believed that if the non-intrusion data is removed from the training data, this will significantly reduce the FP rate. The results reveal that MLP is suitable to perform intrusion detection but further developments and improvements are required for it to be suitable for commercial use.

Similar work was conducted by Hofmaan et al in [38]; they ran experiments for the MLP algorithm on the Lincoln DARPA intrusion detection dataset. In their results it was revealed that on a 5-fold cross validation dataset, the MLP had an overall high classification accuracy rate. Similar results were revealed by the MLP experiments conducted for this dissertation.

In our case study results, the results revealed that the NaiveBayes algorithm’s performance was slightly below that of the MLP throughout all the test cases. Similar results were revealed by Hall et all in [34] when they applied a Bayesian filtering to an intrusion detection system. In their experiments they tested the ability of the Bayesian filter to correctly classify MAC address spoofing. The Bayesian filter achieved a 94-100% success rate, similar to that experienced in our Wi-Fi network anomaly and aberrant behaviour detection experiments.

Panda et all in [65] implemented the NaiveBayes algorithm to the KD99 intrusion detection dataset. In their results for Denial of Service attacks, t was found that the NaiveBayes had a

99% precision rate and a 99.5% recall rate. These results are slightly similar to those revealed by the Wi-Fi Network experiments in this dissertation.

The authors in [1] ran experiments to compare artificial immune systems algorithms, including the AIRS2. The algorithms were applied to a dataset of API call traces of real malware and benign processes running on a Windows operating system. The algorithms were tested for their ability to correctly classify malware and benign processes. In their experiment results the AIRS had a very good performance with average true positive rate of 95%, a false positive rate of 0.83%, recall of 98.5% and an F-measure of 96.4%.

The results from the experiments in [1] display very good algorithm performance, and they are better from what was revealed by the AIRS2 in the Wi-Fi network case studies experiments for this dissertation. The results in the dissertation were poor when compared to those discovered by authors in [1]. This can be attributed to the difference in AIRS2 parameters that were used in the experiments. In the experiments conducted by authors in [1], the 'Affinity Threshold Scalar' was set to 0.4, while in our experiments it was set to 0.2. If the Affinity Threshold Scalar is low, there will be a low replacement rate of best matching memory cells by candidate memory cells, and the reverse is true for higher affinity threshold scalar. For our experiments, there needs to be further investigation to determine which affinity threshold scalar produces better prediction performance for the AIRS2 algorithm.

5.6. Conclusion

5.6.1. Anomaly Network Performance Detection with Network Intrusion

The statistical tests conducted in the anomaly experiments above reveal that in most performance measures there is a significant difference in mean performance for the three algorithms. The percent correct classification, Kappa Statistic, TP Rate and Recall between the MLP and NaiveBayes, there was not a significant difference in mean performance. Therefore we can conclude that the performance measures for all three algorithms are statistically different.

When the bar charts in figure 5.1 to figure 5.7 are carefully examined, one finds out that the three algorithms had an impressive excellent accuracy measures. Across all performance measures, on average the MLP and NaiveBayes had an average anomaly performance above 95%, while the AIRS2 algorithm had an average anomaly performance slightly below 90%.

The bar charts in section 3.1 of the paper reveal that the anomaly data mining capabilities of the three algorithms are closely related. Observing the Figures 5.15 to Figure 5.17, the MLP, NaiveBayes and AIRS2 algorithms have identified similar and closely related detectors at a similar detection rate.

The three algorithms have proven to be able to correctly classify anomaly performance in an intruded network. The MLP was favoured as it had the highest accuracy based on the performance measures and statistical tests presented in section 1 of the Results chapter.

5.6.2. Aberrant Network Performance Detection with Intrusion

In section 2 of the Results chapter the statistical test conducted reveals that there is a statistical difference in mean algorithm performance between the three algorithms. Therefore, one can safely conclude that for aberrant performance detection in an intruded network, the classification accuracy of the MLP, NaiveBayes and AIRS2 algorithms is statistically different.

When the bar charts in figures 5.8 to figure 5.14 are carefully examined, it is found out that the MLP and NaiveBayes, on average, had a mediocre performance for aberrant behaviour detection in an intruded network. Across all performance measures, on average, the performance of MLP and NaiveBayes was below 80% to 65%. Across all performance measures, the average performance of the AIRS2 algorithm was below 40%. The AIRS2 algorithm's performance measures have shown poor performance for aberrant behaviour detection with network intrusion.

The three algorithms have shown mediocre to poor ability in correctly identifying and classifying aberrant behaviour in an intruded network of Wi-Fi hotpots.

5.6.3. Comparing Situation Recognition Before and After Network Intrusion

This section of the dissertation compares performance of the three machine learning algorithms on the RedButton network before, and after intrusion was introduced in the network. The detection performance is compared for anomaly performance detection, aberrant performance detection and the data mining using confusion matrices.

There are qualitative and quantitative methods available to compare performance for the three algorithms on the two case studies; this includes statistical models for hypothesis testing, ANOVA tests, Student t-tests and F-tests. For the purposes of this dissertation, we will use explanatory data analysis as a qualitative descriptive method to analyse the main characteristics of the two case study results, without formulating a hypothesis. We will visually examine the results based on Figures 4.5 to 4.24, and Figures 5.1 to 5.20.

5.6.3.1. Anomaly Performance Detection

For anomaly performance detection before intrusion was introduced in the network, the NaiveBayes and MLP had an average performance level above 98.00% for all performance

measures and an anomaly False Positive rate of less than 2.00%. The AIRS2 machine learning algorithm had an average performance that was below 95.00% across all performance measures and a 19.93% False Positive rate. This was the performance of the machine learning algorithms before intrusion. This was shown by Figures 4.5 to 4.11.

Figure 5.1 to figure 5.8 indicate that when intrusion detection was introduced in the network, the three machine learning algorithms had a slightly different performance. The NaiveBayes and MLP had an average performance that was above 95.00% across all performance measures, with an anomaly False Positive rate below 0.28%. The AIRS had an average performance that was below 93.00% for all performance measures, with a 12.23% anomaly false positive rate.

When we carefully examine the figures 4.21 to figure 4.23, The NaiveBayes and MLP were able to identify similar detectors: '113', '123', '131', '213', '231', '311', '313', but with the NaiveBayes identifying more than those the MLP was unable to identify. The AIRS2 was able to identify fewer anomaly detectors before intrusion, relative to the MLP and NaiveBayes. The detection rates for all detectors were slightly different and varied.

When intrusion was introduced to the network, confusion matrices were again used for performance measuring. The MLP was able to identify the detectors '113', '213', '313', '323', '333', and the NaiveBayes identified similar detectors, but with an extra '311' detector identified. The AIRS2 identified similar detectors as those in NaiveBayes, but with an extra '213'detector identified. Using graphical methods to analyse performance, the detection rate of the three algorithms were slightly similar. This is indicated by figures 5.17 to figure 5.20.

The performance measure comparison results indicate that with intrusion detection introduced in the network, the accuracy of the machine learning algorithms has generally slightly decreased.

5.6.3.2. Aberrant Performance Detection

Figures 4.12 to 4.21 indicate that for aberrant performance detection before intrusion was introduced in the network, the NaiveBayes and MLP had an average performance level below 75.00% for all performance measures and a poor aberrant False Positive rate of more than 58.00%. The AIRS2 machine learning algorithm had an average performance that was below 60.00% across all performance measures and an impressive average 2.00% False Positive rate.

When intrusion detection was introduced in the network, the three machine learning algorithms had a slightly improved performance. The NaiveBayes and MLP had an average performance that was above 75.00% across all performance measures, with a poor aberrant False Positive rate above 55.00%. The AIRS had an average performance that was above 55.00% for all performance measures, with an 11.00% anomaly false positive rate. This is indicated by Figures 5.9 to 5.16.

Before intrusion was introduced in the network, The MLP and NaiveBayes were able to identify similar detectors with the exception of the '113', '233, and '313' detectors. The bar charts in Figures 4.24 and 4.25 indicate that the MLP and NaiveBayes had a slightly similar detection rate, with the '313' detector having an average detection rate 95.07%. On the other hand, the AIRS2 algorithm was able to identify 16 detectors with and addition of eight more to those identified by the MLP and NaiveBayes algorithms. The AIRS2's detection rate for the detectors across all test cases was volatile, this can be explained by the poor performance measures that the AIRS2 displayed.

After intrusion, MLP and NaiveBayes were able to identify similar detectors: '331', '313', '321', '323', '331', '333'. Figures 5.20 and 5.21 indicate that the '313' had an average detection rate of 96.33% for both MLP and NaiveBayes. The AIRS2 was able to identify the same detectors as those of the MLP and NaiveBayes, but the AIRS2 was able to identify six more detectors. The bar chart in figure 5.22 indicates that the detector '313' had a maximum detection rate of 81.04%. This is slightly similar to those shown by the MLP and NaiveBayes machine learning algorithms.

When intrusion was introduced into the network, the aberrant performance of the MLP and NaiveBayes declined when compared their performance before intrusion detection. The AIRS2's performance has worsened but it has drastically improved its aberrant false positive rate performance. The confusion matrices indicate that the aberrant performance of the AIRS2 has slightly improved when compared to the case study before intrusion detection was introduced, but the AIRS2 was unable to outperform the MLP and NaiveBayes.

6. TROMPSBURG DROUGHT MONITORING CASE STUDY

6.1. Introduction

Drought is a natural, environmental disaster that can be classified together with earthquakes, epidemics and floods. It has substantial impact on humans and its impact can persist for several years. A current example of an extreme drought case is that of East Africa. East Africa has experienced its worst drought in 60 years, affecting more than 11 million people. It was declared famine-stricken region, with overcrowded refugee camps in Kenya and Ethiopia. Livestock is dying at a rapid rate, with one farmer reporting have lost 17 goats in one day. Officials also warn that over 800 000 children could die from malnutrition across East Africa nations of Somalia, Ethiopia, Eritrea and Kenya [80].

The definition of drought has been a stumbling block for drought monitoring and analysis, as drought means various things to various people. In general, drought is a deficiency of precipitation relative to what is expected, when extended over a season or a longer period of time results in the inability to meet the demands of human activities and the environment [36]. The impact of drought is big, it "impacts both surface and groundwater resources and can lead to reduced water supply, deteriorated water quality, crop failure reduced range productivity, diminished power generation, disturbed riparian habitats, and suspended recreation activities, as well as a host of economic and social activities" [61].

Given the impacts of drought, there is a need for developing strategies for drought monitoring and early warning systems that are able to determine drought severity. These systems can help in planning and managing water resources, and can help reduce and avoid impacts of drought.

In this study we investigate available drought monitoring tools, and look into the use of machine learning algorithms in drought forecasting. Three machine learning algorithms, viz. Artificial Immune Systems, Bayesian Networks and Artificial Neural Networks, are studied and their performance on a South African precipitation dataset is compared.

6.2. Background

6.2.1. Definitions of drought and its classifications

Drought impacts different people in different ways, and for this matter it has been classified into four categories:

- Meteorological drought: This is the general concept of drought; it is defined as the lack of precipitation or rainfall in a particular region over a period of time.
- Hydrological drought: This refers to a prolonged period of inadequate surface and subsurface water resources in lakes, dams, natural reservoir and soil.

- Agricultural drought: This refers to a prolonged lack of precipitation that leads to declined soil moisture that adversely affects crop production.
- Socio-economic drought: This is drought-associated to the demand and supply of economic good, water, when available water resource systems are unable to meet the demand for water.

6.2.2. Drought Indices

Research in previous years has developed indices that measure drought. These indices can be used in early warning and drought monitoring systems. These indices are very important in measuring the drought severity, intensity, duration, coverage and magnitude. These indices are discussed in the section below.

6.2.2.1. Percent Normal Precipitation

This index is calculated using precipitation. It is a percentage ratio of current precipitation to a (normally) 30-year average precipitation [32]. It falls short in that it does not allow comparison between different locations [86].

6.2.2.2. Crop Moisture Index

This is an index used in crop-producing regions. It evaluates short-term moisture conditions by using weekly values of temperature and precipitation. It has shortfalls in that it has unnatural changes in temperature due "to the dependence of the abnormal evapotranspiration term on the magnitude of potential evapotranspiration" [61]. It is not good for long-term drought monitoring as its rapid responses to changing short-term conditions provide misleading information about long-term conditions [61, 86].

6.2.2.3. Standard Precipitation Index

The standard Precipitation Index (SPI) is a probability index that is calculated using long-term precipitation. It can be calculated on different time scales allowing both short-term (soil moisture) and long-term (water resource systems) monitoring. Details of this drought monitoring index are discussed in the next section.

6.2.2.4. Palmer Drought Severity Index

The Palmer Drought Severity Index (PDSI) is used to measure drought severity over a long-term period. It is calculated using three variables, viz. temperature, soil moisture and precipitation. It has been well tested and in use for a long period of time, and allows comparison of different climatic zones. According to [61], the rules used to establish PDSI's are arbitrary and its limitations have been documented in several studies. "Limitations of PDSI include: (1) an inherent time scale making PDSI more suitable for agricultural impacts and not so much for hydrologic droughts, (2) assumptions that all precipitation is rain, thus making values during winter months and at high elevations often questionable. PDSI also assumes that run-off only occurs after all soil layers have become saturated, leading to an

underestimation of runoff, and (3) PDSI can be slow to respond to developing and diminishing droughts" [61].

6.2.2.5. Conclusion

The authors in [61] have made great attempts to make comparisons and find out which of the drought indices are most suitable for drought monitoring. They ran a workshop in which delegates over across 22 countries were to develop a consensus standard index for each type of drought. This was in the efforts to develop guidelines to help explore drought indices that may have global applications. In this workshop, they listed the relevant indices and evaluated them according to the following criteria:

- Regions were the indices are used.
- The type of drought being monitored.
- How indices are used.
- Advantages and disadvantages.
- Overall general usefulness.

For most of the discussions and work presented at this workshop, (more specifically) the SPI and PDSI was the most recommended for monitoring.

The comparison between SPI and PDSI came out with the following results:

i. "Special characteristics of PDI vary from site to site while those of SPI do not vary from site to site. Also, PDI has a complex structure with an exceptionally long memory, while SPI is an easily interpreted, simple moving average process. Therefore, SPI can be used as the primary drought index, because it is simple, spatially invariant in its interpretation, and probabilistic, so it can be used in risk and decision analysis" [32].
ii. "SPI is more representative of short-term precipitation than PDSI and thus is a better indicator for soil moisture variation and soil wetness" [76].
iii. "SPI is a better predictor of crop production, as it represents the moisture state of soil better" [68].
iv. "SPI provides a better spatial standardization than does PDSI with respect to extreme drought events" [52].
v. It was found out that the SPI was a valuable estimator for drought severity [61].
vi. SPI detects the onset of a drought earlier than PDSI [35].

Based on this, it can be inferred that the SPI is a better monitoring tool to use. We will therefore focus on the SPI for the remainder of this case study. The section that follows will give details and a description of how the SPI is calculated.

6.3. Standard Precipitation Index (SPI)

6.3.1. Background of SPI

The SPI was developed by McKee *et al*. in 1993[56]. The standard Precipitation Index (SPI) is a probability index that is calculated using long-term precipitation. It can be calculated on different time scales allowing both short-term (soil moisture) and long-term (water resource systems) monitoring. It is a simple drought monitor index tool that takes in one parameter for its computation, long term precipitation data. "This long-term record is fitted into a probability distribution, which is then transformed to a normal distribution so that the mean SPI for the location and desired period is zero [56]. Its strength lies in the fact that it can be calculated for different time scales, from $3-24$ months. This allows one to monitor and use for different domains. The authors in [61] talks about the different time scales for different domains of drought monitoring, saying that SPI can "monitor short-term water supplies, such as soil moisture which is important for agricultural production, and long-term water resources, such as groundwater supplies, stream flow, and lake and reservoir levels. Soil moisture conditions respond to precipitation anomalies on a relatively short scale. Groundwater, stream flow, and reservoir storage reflect the long-term precipitation anomalies".

SPI values have range -3 to 3. Positive SPI values indicate greater than median precipitation, and negative values indicate less than median precipitation. Because the SPI is normalised, wetter and drier climates can be represented in the same way, and wet periods can also be monitored using the SPI. The table 6.1 below represents the values of SPI and how they are classified, and the percentage occurrence.

Table 6.1: SPI Classification and Corresponding Probabilities of Occurrence [56, 57]

SPI Value	Drought Category	Occurrence
2.00 and above	Extremely Wet	2.3%
1.50 to 1.99	Very Wet	4.4%
1.00 to 1.49	Moderately Wet	9.2%
-0.99 to 0.99	Near Normal	68.2%
-1.00 to -1.49	Moderately Dry	4.4%
-1.50 to -1.99	Severely Dry	9.2%
-2.00 and less	Extremely Dry	2.3%

6.3.2. Calculating SPI

SPI is uniquely related to probability; it is calculated by "fitting a probability density function to the frequency distribution of precipitation summed over a time scale of interest. Each probability density function is the transformed into the standardised normal distribution" [60]. The probability density function of the gamma distribution is:

$$g(x) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{-x/\beta} \; for \; x > 0$$

Where $\alpha > 0$ represents the shape parameter; $\beta > 0$ is a scale parameter, and $x > 0$ is the amount of precipitation. $\Gamma(\alpha)$ is the gamma function, which is defined as [40, 40, 40, 52]:

$$\Gamma(\alpha) = \lim_{n\to\infty} \prod_{v=0}^{n-1} \frac{n!\, n^{y-1}}{y+v} \equiv \int_0^{\infty} y^{\alpha-1} e^{-y}\, dy$$

The two parameters must be estimated and to get the optimum parameters, the maximum likelihood method is used:

$$\hat{\alpha} = \frac{1}{4A}\,(1 + \sqrt{1 + \frac{4A}{3}}$$

$$\hat{\beta} = \frac{\bar{x}}{\hat{\alpha}}$$

Where for n observations

$$A = \ln(\bar{x}) - \frac{\sum \ln(x)}{n}$$

This process is repeated until the algorithm converges. Integrating the probability function yields an expression for the cumulative probability $G(x)$ of an observed amount of precipitation occurring for a given month and time scale:

$$G(x) = \int_0^x g(x)\, dx = \frac{1}{\hat{\beta}^{\hat{\alpha}}\Gamma(\alpha)} \int_0^x x^{\hat{\alpha}} e^{-x/\hat{\beta}}\, dx$$

When substituting t for $x/\hat{\beta}$ the equation above is reduced to an incomplete gamma function:

$$G(x) = \frac{1}{\Gamma(\hat{\alpha})} \int_0^x t^{\hat{\alpha}-1} e^{-1}\, dt$$

Since the gamma distribution is undefined for values $x = 0$, and a distribution may contain zeroes, $q = P(x = 0) > 0 \; where \; P(x = 0)$ is the probability of zero precipitation, the cumulative probability becomes:

$$H(x) = q + (1 - q)G(x)$$

The cumulative probability $H(x)$ is then transformed into a standard normal random variable with mean zero and variance one, which is the value of SPI.

6.4. Related Work

There has been creditable work done to predict weather condition using Bayesian Networks, and in [46] they were applied to the problem of predicting sea breeze. Bayesian Networks were then compared with existing rule-based system and it was found that the Bayesian network outperformed the traditional rule-based system in prediction accuracy.

The authors in [4] introduced a Bayesian Network framework that deals with multivariate spatiality distributed time series. They used it to predict precipitation for 100 stations in the North basin of the Iberian Peninsula during winter of 1999. In [19], Bayesian networks are used to estimate forecasts of peak and average temperatures. In this case study, data derived from a power utility system is used to forecast electric load with imperfect information.

Considerable weather forecast work has focused on the use of artificial neural networks and Bayesian Networks, but only a few use artificial immune systems for weather forecasting. The authors in [93] implemented an immune-based algorithm that was applied on weather data for forecasting. The immune algorithm was compared to an artificial neural network algorithm and the results reveal that the implemented immune algorithm had a higher forecast accuracy rate than that of the neural network.

There has also been great work done in drought monitoring using artificial neural networks. [3] used feed-forward ANN with multilayer perceptions (MLP) for empirical model development using seven climatic variables (monthly mean air temperature, monthly mean daily minimum and maximum air temperature, monthly mean relative humidity, monthly precipitation, monthly mean global solar irradiation and monthly potential evapotranspiration).

Authors in [60] used a record of SPI time series data and linear stochastic models (ARIMA/SARIMA), recursive multistep neural networks (DMSNN) for drought forecasting in the Kansabiti river basin, which lies in the Purulia district of West Bengal, India. In their comparison they found neural networks to be more suitable for drought forecasting. The authors in [73] used a temporal back-propagation neural network (TBP-NN) for monthly rainfall-runoff modelling in scarce data conditions.

In this study, we would like to investigate the performance of the three machine learning algorithms in the aim to answer the following questions:

- Which method performs better?
- How do the methods perform across different SPI time scales?
- What kind of information can be mined using these machine learning algorithms?

6.5. Research Methods

6.5.1. Drought Monitoring Region and Data Collection

The region of monitoring is Trompsburg, Free State, South Africa. Trompsburg is a small town located in the southern Free State. It is in the ecotone between Nama-Karoo and the grassland biome. The main land use in the region is livestock farming, especially sheep and cattle farming. Frequency of drought is predicted to increase in that region. Annual rainfall for the last 95 years has remained unchanged. Weather monitoring data was collected from this region by the South African Weather Services, but for the purposes of this study, the main focus was on this regions precipitation. Monthly precipitation data for this region was collected for the years January 1913 to May 2009, making a total of 96 years of observations.

6.5.2. Test Cases

In the literature reviewed above, it was found that there are four different types of drought: meteorological, hydrological, agricultural, and socio-economic droughts. Using Standard Precipitation Index (SPI) allows the monitoring of different types of drought using different time scales [40, 56, 57, 60, 61].

This case study will focus on the following time scales: SPI 3 months, SPI 6 months, SPI 12 months and SPI 24 months for the Trompsburg region.

6.5.3. Experiment Design

The data provided for the Trompsburg region was precipitation data. We had to transform and calculate the data into precipitation data in such a way that we can create training and testing dataset. The steps taken for completing the experiments in this study are shown below by figure 6.1.

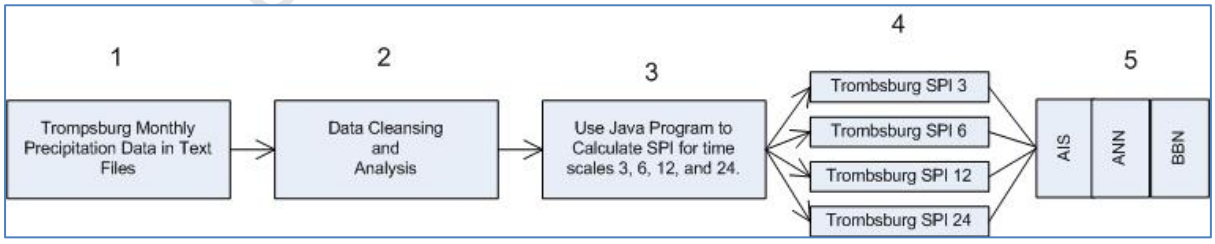

Figure 6.1: Trompsburg Experiment

1. The data is prepared into text files for further processing
2. Data cleansing and Analysis: Data is checked for any missing data. If there is missing data, the average precipitation is used.

3. A Java program that is designed for calculating the monthly SPI when given time-scale inputs and monthly precipitation data. This program can be found and downloaded from the Greenleaf website[1].
4. The output files are produced by the Java Program for time scales 3, 6, 12 and 24.
5. The data is then used by the Weka program to test performance measures using the AIS, ANN and BNN algorithms.

In Weka, the 10-fold cross validation technique was used, and for each of the four test cases, the experiments were iterated 10 times. The mean algorithm performance measures of each experiment's iteration were recorded and used for comparison. In total we had 400 observations for statistical comparisons. The results for the Trompsburg drought experiment follows in the section below. In the evaluation, bar charts graphical techniques and t-test statistics are used.

6.6. Experiment Results

The results that follow are from a study conducted to assess the ability of the algorithms MLP, NaiveBayes and AIRS2 for data mining on a South African Standard Precipitation Index drought case study. The algorithms are assessed based on the following algorithm performance measures: Kappa Statistic, True Positive Rate, False Positive Rate, Precision, Recall, F-measure, ROC Area and a confusion matrix.

The chapter presents statistical hypothesis test which were conducted for each algorithm performance measure to investigate statistical differences in algorithms performance.

[1] http://greenleaf.unl.edu/downloads/

6.6.1. Kappa Statistic Performance

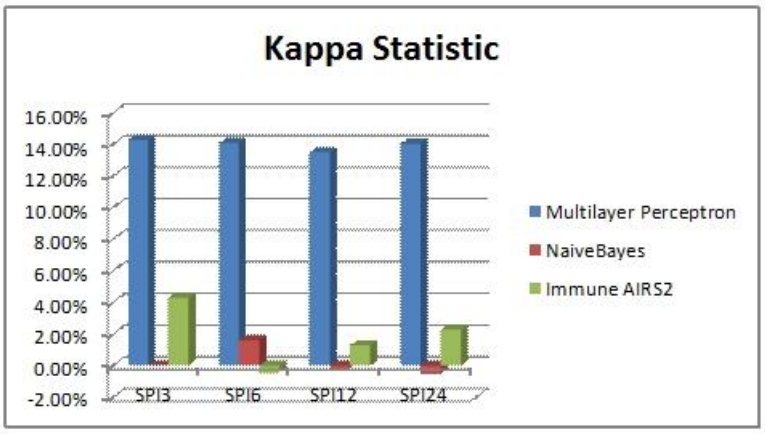

Figure 6.2: SPI Kappa Statistic Performance Bar Chart

Across all SPI test cases, the MLP had an average Kappa statistic of 13.99%, while that of the NaiveBayes was 0.22%. The NaiveBayes had a below zero Kappa statistic for SPI12 and SPI24 test cases, this is indicated by Figure 6.3 above. The AIRS2 had an average kappa statistic of 2.28%, and in the test case SPI6, the Kappa statistic was -0.45%.

Table 6.2: Results for Kappa Statistic T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.005186847	0.002797439
Variance	0.000582455	0.00039208
Observations	400	400
Pearson Correlation	0.333264403	
Hypothesized Mean Difference	0	
df	399	
t Stat	1.86579402	
P(T<=t) one-tail	0.031401884	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	0.062803768	
t Critical two-tail	1.965927296	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.005186847	0.00679047
Variance	0.000582455	0.003299065
Observations	400	400
Pearson Correlation	0.01119336	
Hypothesized Mean Difference	0	
df	399	
t Stat	-0.516861807	
P(T<=t) one-tail	0.3027695	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	0.605539	
t Critical two-tail	1.965927296	

For the test:

Kappa Statistic: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Kappa Statistic: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 1.865 and its two-tailed p-value is 0.0628. At 5% confidence level, the test is not statistically significant and there is weak evidence to infer that the alternative hypothesis is true. Therefore we do not reject the null hypothesis and conclude that the difference in the mean Kappa statistic for the MLP and NaiveBayes algorithms equals zero, the hypothesised mean.

For the test:

Kappa Statistic: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Kappa Statistic: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is -0.516 and its two-tailed p-value is 0.605. At 5% confidence level, the test is not statistically significant and there is little to no evidence to infer that the alternative hypothesis is true. Therefore we do not reject the null hypothesis and conclude that the difference in mean Kappa statistic for the MLP and AIRS2 algorithms equals zero, the hypothesised mean.

6.6.2. True Positive Rate Performance

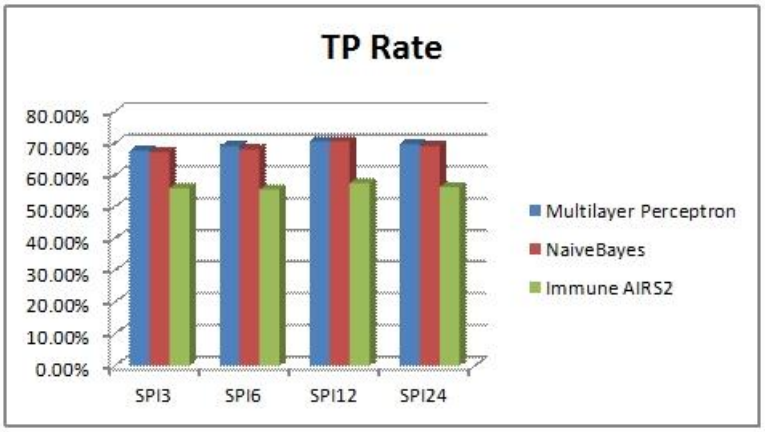

Figure 6.3: SPI True Positive Rate Performance Bar Chart

Figure 6.4 indicates that across all test cases, the MLP had an average true positive rate of 69.38%, while the NaiveBayes' was slightly lower at 68.75%. The AIRS2 had an even lower average true positive rate performance of 56.30%. Amongst all time scales, the time scale for SPI12 had the high performance for all algorithms. The MLP had a performance of 70.70% and the NaiveBayes had a performance of 70.60% while the AIRS2 had a performance of 57.50%.

Table 6.3: Results for True Positive Rate T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.685879685	0.68782946
Variance	0.000345877	0.000237031
Observations	400	400
Pearson Correlation	0.611822971	
Hypothesized Mean Difference	0	
df	399	
t Stat	-2.557183544	
P(T<=t) one-tail	0.005460679	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	0.010921357	
t Critical two-tail	1.965927296	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.685879685	0.554753373
Variance	0.000345877	0.001545385
Observations	400	400
Pearson Correlation	0.008462227	
Hypothesized Mean Difference	0	
df	399	
t Stat	60.50194735	
P(T<=t) one-tail	2.1214E-203	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	4.2428E-203	
t Critical two-tail	1.965927296	

For the test:

True Positive Rate: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

True Positive Rate: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -2.557 and its two-tailed p-value is 0.0109. At 5% confidence level, the test is significant and there is strong evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean true positive rate for the MLP and NaiveBayes algorithms.

For the test:

True Positive Rate: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

True Positive Rate: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 60.501 and its two-tailed p-value is 4.2428E-203. At 5% confidence level, the test is significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in mean true positive rate for the MLP and AIRS2 algorithms.

6.6.3. False Positive Rate Performance

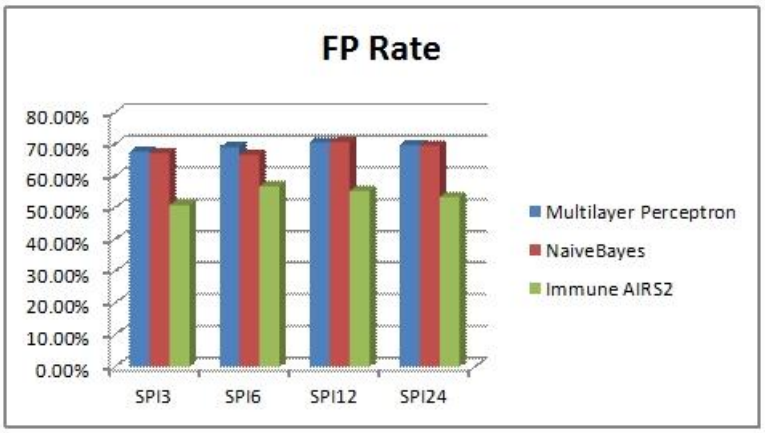

Figure 6.4: SPI False Positive Rate Performance Bar Chart

The average Positive Rate for the MLP was 69.38% across all SPI test cases, while that of the NaiveBayes was slightly lower at 68.63%. The AIRS2 had an impressive lower average false positive rate performance of 54.15%. This is shown by the bar chart in Figure 6.5.

Table 6.4: Results for False Positive Rate T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.682129435	0.686169001
Variance	0.000556912	0.000388558
Observations	400	400
Pearson Correlation	0.585725143	
Hypothesized Mean Difference	0	
df	399	
t Stat	-4.036870174	
P(T<=t) one-tail	3.2478E-05	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	6.4956E-05	
t Critical two-tail	1.965927296	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.682129435	0.548228652
Variance	0.000556912	0.002574156
Observations	400	400
Pearson Correlation	0.134442228	
Hypothesized Mean Difference	0	
df	399	
t Stat	50.52746045	
P(T<=t) one-tail	8.6434E-176	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	1.7287E-175	
t Critical two-tail	1.965927296	

For the test:

False Positive Rate: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

False Positive Rate: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -4.036 and its two-tailed p-value is 6.4956E-05. At 5% confidence level, the test is significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean false positive rate for the MLP and NaiveBayes algorithms.

For the test:

False Positive Rate: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

False Positive Rate: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 50.527 and its two-tailed p-value is 1.728E-175. At 5% confidence level, the test is significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in mean false positive rate for the MLP and AIRS2 algorithms.

6.6.4. Precision Performance

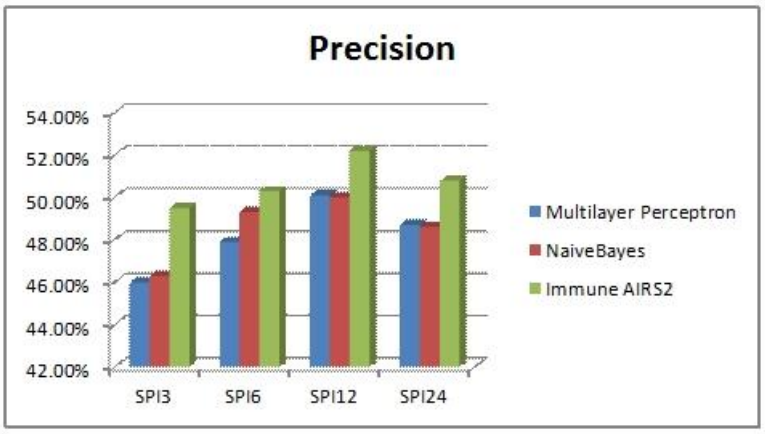

Figure 6.5: SPI Precision Performance Bar Chart

The bar chart in Figure 6.6 indicates that the MLP had an average precision performance of 48.18%, while that of the NaiveBayes was slightly higher at 48.55%, across all test cases. The average precision rate performance of the AIRS2 was 50.70%, slightly higher than that of both the MLP and NaiveBayes algorithms. The bar chart in Figure 6.6 indicates, for all algorithms, there are increasing trend in precision, from test case SPI3-SPI12, then a sudden decrease in precision for test case SPI24. For all time scales, the precision performance for both the MLP and NaiveBayes was below 50% while that of the AIRS2 was slightly above 50%. The time scale SPI12 had a better performance over other time scales.

Table 6.5: Results for Precision T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.490778328	0.4857684
Variance	0.000628626	0.00033958
Observations	400	400
Pearson Correlation	0.520003468	
Hypothesized Mean Difference	0	
df	399	
t Stat	4.537191917	
P(T<=t) one-tail	3.77691E-06	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	7.55381E-06	
t Critical two-tail	1.965927296	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.490778328	0.505067519
Variance	0.000628626	0.000900789
Observations	400	400
Pearson Correlation	0.177117354	
Hypothesized Mean Difference	0	
df	399	
t Stat	-8.041948277	
P(T<=t) one-tail	5.08513E-15	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	1.01703E-14	
t Critical two-tail	1.965927296	

For the test:

Precision: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Precision: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 4.537 and its two-tailed p-value is 7.55381E-06. At 5% confidence level, the test is significant and there is overwhelming evidence to infer that the

alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean precision for the MLP and NaiveBayes algorithms.

For the test:

Precision: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Precision: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is -8.041 and its two-tailed p-value is 1.01703E-14. At 5% confidence level, the test is significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in mean precision for the MLP and AIRS2 algorithms.

6.6.5. Recall Performance

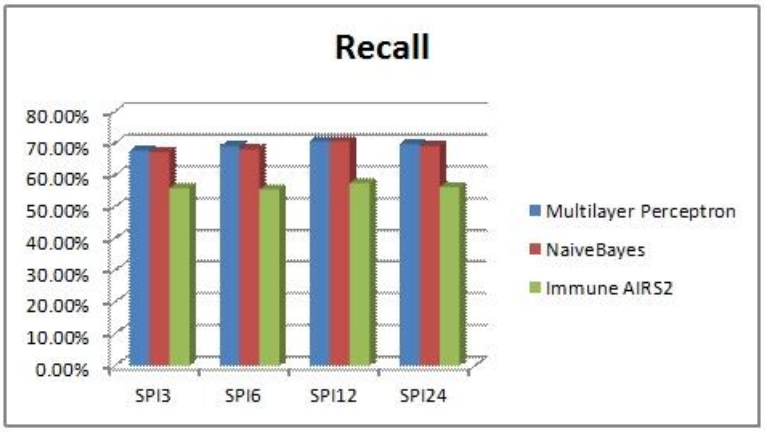

Figure 6.6: SPI Recall Performance Bar Chart

The average SPI recall performance of the MLP was 69.38%, across all test cases, while that of the NaiveBayes was slightly lower at 68.75%. The AIRS2 had an average SPI recall performance of 56.30%. This is shown by the bar chart in figure 6.7. Amongst all SPI time scales, classifications for the SPI12 produced higher recall performance for all algorithms. The MLP, NaiveBayes and AIRS2 had performance of 70.70%, 70.60% and 57.50%, respectively, for the SPI12 time scale.

Table 6.6:Results for Recall T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.685879685	0.68782946
Variance	0.000345877	0.000237031
Observations	400	400
Pearson Correlation	0.611822971	
Hypothesized Mean Difference	0	
df	399	
t Stat	-2.557183544	
P(T<=t) one-tail	0.005460679	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	0.010921357	
t Critical two-tail	1.965927296	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.685879685	0.554753373
Variance	0.000345877	0.001545385
Observations	400	400
Pearson Correlation	0.008462227	
Hypothesized Mean Difference	0	
df	399	
t Stat	60.50194735	
P(T<=t) one-tail	2.1214E-203	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	4.2428E-203	
t Critical two-tail	1.965927296	

For the test:

Recall: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

Recall: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -2.557 and its two-tailed p-value is 0.0109. At 5% confidence level, the test is significant and there is strong evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean recall for the MLP and NaiveBayes algorithms.

For the test:

Recall: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

Recall: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 60.501 and its two-tailed p-value is 4.2428E-203. At 5% confidence level, the test is significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in mean recall for the MLP and AIRS2 algorithms.

6.6.6. F-Measure Performance

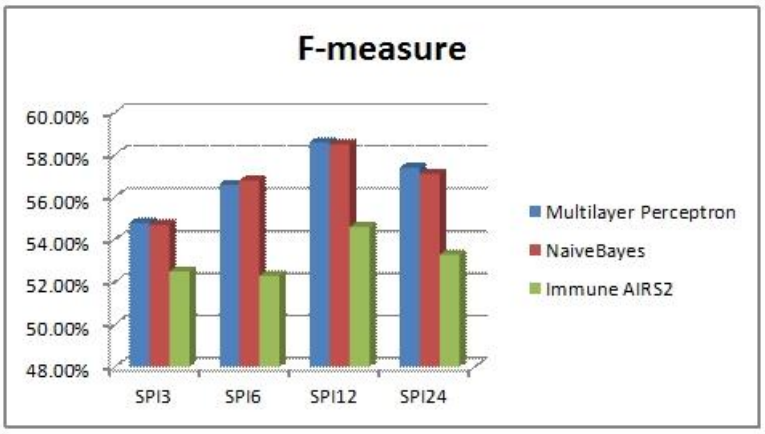

Figure 6.7: SPI F-Measure Performance Bar Chart

The bar chart in figure 6.8 indicate that the MLP had an average SPI F-measure performance of 56.85%, and that of the NaiveBayes was slightly lower at 56.78%. The AIRS2 algorithm had a lower average SPI F-measure performance of 53.18%. Figure 6.8's bar chart again indicates a trend that, for all algorithms, from test cases SPI3-SPI12, there is an increasing or improving f-measure performance, but there is a sudden decline in f-measure performance for test case SPI24. When compared to other times scales. The SPI12 time scale produced higher F-measure performance. The MLP, NaiveBayes and the AIRS2 had a performance of 58.60%, 58.50% and 54.60%, respectively, for the SPI12 time scale.

Table 6.7: Results for F-Measure T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.568638974	0.567942917
Variance	0.000273604	0.000243007
Observations	400	400
Pearson Correlation	0.824673061	
Hypothesized Mean Difference	0	
df	399	
t Stat	1.456742765	
P(T<=t) one-tail	0.072986941	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	0.145973883	
t Critical two-tail	1.965927296	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.568638974	0.525369201
Variance	0.000273604	0.000857692
Observations	400	400
Pearson Correlation	0.224201547	
Hypothesized Mean Difference	0	
df	399	
t Stat	28.62350407	
P(T<=t) one-tail	4.694E-99	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	9.3881E-99	
t Critical two-tail	1.965927296	

For the test:

F-Measure: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

F-Measure: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is 1.456 and its two-tailed p-value is 0.146. At 5% confidence level, the test is not statistically significant and there is little or no evidence to infer that the alternative hypothesis is true. Therefore we do not reject the null hypothesis and conclude

that the difference in the mean recall for the MLP and NaiveBayes algorithms equals zero, the hypothesised mean.

For the test:

F-Measure: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

F-Measure: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 28.623 and its two-tailed p-value is 9.3881E-99. At 5% confidence level, the test is significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in mean recall for the MLP and AIRS2 algorithms.

6.6.7. ROC Area

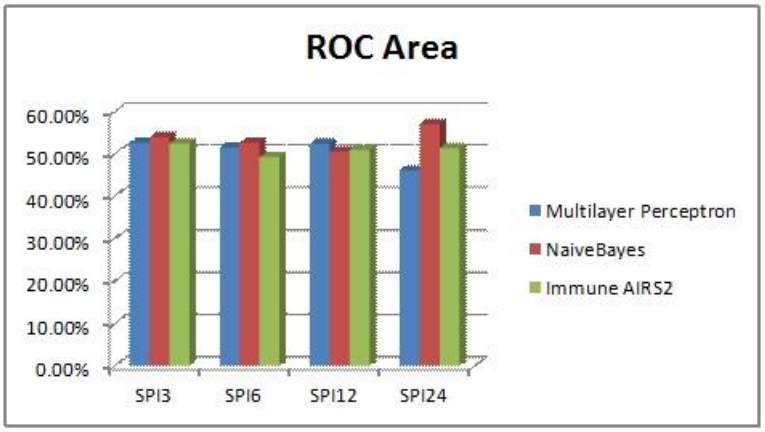

Figure 6.8: SPI ROC Area Performance Bar Chart

The bar chart in figure 6.9 indicates that the average SPI ROC area performance for the MLP was 50.73%, while that of the NaiveBayes was slightly higher at 53.60%. The AIRS2 had an average SPI ROC Area performance of 51.05%. For MLP, the higher ROC Area performance was noticed with time scale SPI12 of 52.40%. The NaiveBayes in SPI 24 had a higher ROC area performance of 57.10%. The AIRS2 had its higher ROC area performance in the time scale SPI3.

Table 6.8: Results for ROC Area T-test for Paired Two Samples for Means

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*NaiveBayes*
Mean	0.532997238	0.544799739
Variance	0.003981866	0.002252882
Observations	400	400
Pearson Correlation	0.62373355	
Hypothesized Mean Difference	0	
df	399	
t Stat	-4.722465905	
P(T<=t) one-tail	1.616E-06	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	3.23199E-06	
t Critical two-tail	1.965927296	

t-Test: Paired Two Sample for Means		
	Multilayer Perceptron	*AIRS2*
Mean	0.532997238	0.50326236
Variance	0.003981866	0.00109709
Observations	400	400
Pearson Correlation	0.164564768	
Hypothesized Mean Difference	0	
df	399	
t Stat	8.974538797	
P(T<=t) one-tail	5.6423E-18	
t Critical one-tail	1.648681534	
P(T<=t) two-tail	1.12846E-17	
t Critical two-tail	1.965927296	

For the test:

ROC Area: H_0: $\mu_{MLP} - \mu_{NaiveBayes} = 0$

ROC Area: H_1: $\mu_{MLP} - \mu_{NaiveBayes} \neq 0$

The value of the t-Statistic is -4.722 and its two-tailed p-value is 3.23199E-06. At 5% confidence level, the test is significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in the mean recall for the MLP and NaiveBayes algorithms.

For the test:

ROC Area: H_0: $\mu_{MLP} - \mu_{AIRS2} = 0$

ROC Area: H_1: $\mu_{MLP} - \mu_{AIRS2} \neq 0$

The value of the t-Statistic is 8.974 and its two-tailed p-value is 1.12846E-17. At 5% confidence level, the test is significant and there is overwhelming evidence to infer that the alternative hypothesis is true. Therefore we reject the null hypothesis and conclude that there is a difference in mean recall for the MLP and AIRS2 algorithms.

6.6.8. Confusion Matrix

The section that follows uses confusion matrices to assess the ability of the MLP, NaiveBayes and AIRS2 algorithms to make correct classifications. Confusion matrices for all four test cases are discussed.

6.6.8.1. Multilayer Perceptron Algorithm

Table 6.9: Multilayer Perceptron Confusion Matrices for (a) SPI3 and (b) SPI6 Test Cases

Near_Normal	Moderately_Dry	Severely_Dry	Moderately_Wet	Very_Wet	Extremely_Dry	Extremely_Wet	
781	0	0	0	0	0	0	Near_Normal
129	0	0	0	0	0	0	Moderately_Dry
43	0	0	0	0	0	0	Severely_Dry
107	0	0	0	0	0	0	Moderately_Wet
42	0	0	0	0	0	0	Very_Wet
23	0	0	0	0	0	0	Extremely_Dry
27	0	0	0	0	0	0	Extremely_Wet

(a) SPI 3

Near_Normal	Severely_Dry	Moderately_Dry	Moderately_Wet	Extremely_Dry	Very_Wet	Extremely_Wet	
797	0	0	0	0	0	0	Near_Normal
46	0	0	0	0	0	0	Severely_Dry
125	0	0	0	0	0	0	Moderately_Dry
92	0	0	0	0	0	0	Moderately_Wet
23	0	0	0	0	0	0	Extremely_Dry
36	0	0	0	0	0	0	Very_Wet
33	0	0	0	0	0	0	Extremely_Wet

(b) SPI 6

Table 6.10: Multilayer Perceptron Confusion Matrices for (c) SPI12 and (d) SPI24 Test Cases

Near_Normal	Moderately_Dry	Severely_Dry	Moderately_Wet	Very_Wet	Extremely_Wet	Extremely_Dry	
815	0	0	0	0	0	0	Near_Normal
105	0	0	0	0	0	0	Moderately_Dry
59	0	0	0	0	0	0	Severely_Dry
80	0	0	0	0	0	0	Moderately_Wet
25	0	0	0	0	0	0	Very_Wet
42	0	0	0	0	0	0	Extremely_Wet
26	0	0	0	0	0	0	Extremely_Dry

(c) SPI 12

Near_Normal	Moderately_Dry	Moderately_Wet	Very_Wet	Severely_Dry	Extremely_Dry	Extremely_Wet	
804	0	0	0	0	0	0	Near_Normal
94	0	0	0	0	0	0	Moderately_Dry
99	0	0	0	0	0	0	Moderately_Wet
27	0	0	0	0	0	0	Very_Wet
33	0	0	0	0	0	0	Severely_Dry
52	0	0	0	0	0	0	Extremely_Dry
43	0	0	0	0	0	0	Extremely_Wet

(d) SPI 24

When working with a confusion matrix, good results corresponds to large numbers on the main diagonal and small, ideally zero, off-diagonal elements. In table 6.10(a, b) and 6.11 (a, b), the main diagonals elements are zeroes, except for the first column. The column indicates the predicted class, and most predictions fell into the first column class 'Near_Normal'. This is an indication that the MLP algorithm was biased to predicting most instances as 'Near Normal'. This can help understand and explain the poor algorithm performance measures shown by the bar charts in figures 6.2-6.9.

6.6.8.2. NaiveBayes Algorithm

Table 6.11: NaiveBayes Confusion Matrices for (a) SPI3 and (b) SPI6 Test Cases

Near_Normal	Moderately_Dry	Severely_Dry	Moderately_Wet	Very_Wet	Extremely_Dry	Extremely_Wet	
773	0	8	0	0	0	0	Near_Normal
129	0	0	0	0	0	0	Moderately_Dry
42	0	1	0	0	0	0	Severely_Dry
105	0	2	0	0	0	0	Moderately_Wet
42	0	0	0	0	0	0	Very_Wet
23	0	0	0	0	0	0	Extremely_Dry
27	0	0	0	0	0	0	Extremely_Wet

(a) SPI 3

Near_Normal	Severely_Dry	Moderately_Dry	Moderately_Wet	Extremely_Dry	Very_Wet	Extremely_Wet	
781	0	0	14	2	0	0	Near_Normal
46	0	0	0	0	0	0	Severely_Dry
120	0	0	5	0	0	0	Moderately_Dry
87	0	0	4	1	0	0	Moderately_Wet
21	0	0	2	0	0	0	Extremely_Dry
35	0	0	1	0	0	0	Very_Wet
33	0	0	0	0	0	0	Extremely_Wet

(b) SPI 6

Table 6.12: Multilayer Perceptron Confusion Matrices for (c) SPI12 and (d) SPI24 Test Cases

Near_Normal	Moderately_Dry	Severely_Dry	Moderately_Wet	Very_Wet	Extremely_Wet	Extremely_Dry	
813	0	0	0	0	0	2	Near_Normal
105	0	0	0	0	0	0	Moderately_Dry
59	0	0	0	0	0	0	Severely_Dry
80	0	0	0	0	0	0	Moderately_Wet
25	0	0	0	0	0	0	Very_Wet
42	0	0	0	0	0	0	Extremely_Wet
26	0	0	0	0	0	0	Extremely_Dry

(c) SPI 12

Near_Normal	Moderately_Dry	Moderately_Wet	Very_Wet	Severely_Dry	Extremely_Dry	Extremely_Wet	
796	0	0	8	0	0	0	Near_Normal
94	0	0	0	0	0	0	Moderately_Dry
99	0	0	0	0	0	0	Moderately_Wet
27	0	0	0	0	0	0	Very_Wet
33	0	0	0	0	0	0	Severely_Dry
51	0	0	1	0	0	0	Extremely_Dry
43	0	0	0	0	0	0	Extremely_Wet

(d) SPI 24

The tables 6.12(a, b) and 6.13(c, d) are NaiveBayes algorithm's confusion matrices for test cases SPI3 to SPI24 test experiments. The confusion matrices indicate that across the main diagonal are zero values, except for the first column. The first column indicates the predicted class, and for most instances, there NaiveBayes algorithm predicted the class 'Near_Normal'. This serves as a sign that the NaiveBayes algorithm was biased towards predicting a 'Near_Normal' class. This can well help us understand and explain the poor performance measures of the NaiveBayes as shown by the bar charts of Figures 6.2 to 6.9.

6.6.8.3. AIRS2 Algorithm

Table 6.13: AIRS2 Confusion Matrices for (a) SPI3 and (b) SPI6 Test Cases

Near_Normal	Moderately_Dry	Severely_Dry	Moderately_Wet	Very_Wet	Extremely_Dry	Extremely_Wet	
613	105	28	28	2	4	1	Near_Normal
92	24	6	5	0	2	0	Moderately_Dry
27	7	6	2	1	0	0	Severely_Dry
84	13	8	1	1	0	0	Moderately_Wet
26	9	3	4	0	0	0	Very_Wet
20	1	2	0	0	0	0	Extremely_Dry
18	6	2	1	0	0	0	Extremely_Wet

(a) SPI 3

Near_Normal	Severely_Dry	Moderately_Dry	Moderately_Wet	Extremely_Dry	Very_Wet	Extremely_Wet	
619	90	63	19	5	1	0	Near_Normal
36	7	2	1	0	0	0	Severely_Dry
104	12	5	2	2	0	0	Moderately_Dry
70	6	6	8	1	0	1	Moderately_Wet
20	2	1	0	0	0	0	Extremely_Dry
28	4	1	3	0	0	0	Very_Wet
26	5	2	0	0	0	0	Extremely_Wet

(b) SPI 6

Table 6.14: AIRS2 Confusion Matrices for (c) SPI12 and (d) SPI24 Test Cases

Near_Normal	Moderately_Dry	Severely_Dry	Moderately_Wet	Very_Wet	Extremely_Wet	Extremely_Dry	
644	96	37	26	7	2	3	Near_Normal
85	13	4	3	0	0	0	Moderately_Dry
50	7	2	0	0	0	0	Severely_Dry
65	10	2	3	0	0	0	Moderately_Wet
16	2	4	3	0	0	0	Very_Wet
20	12	4	5	1	0	0	Extremely_Wet
20	0	4	1	1	0	0	Extremely_Dry

(c) SPI 12

Near_Normal	Moderately_Dry	Moderately_Wet	Very_Wet	Severely_Dry	Extremely_Dry	Extremely_Wet	
624	97	62	18	1	2	0	Near_Normal
67	21	5	1	0	0	0	Moderately_Dry
81	11	3	3	0	1	0	Moderately_Wet
19	3	5	0	0	0	0	Very_Wet
20	11	2	0	0	0	0	Severely_Dry
43	6	2	1	0	0	0	Extremely_Dry
28	7	4	4	0	0	0	Extremely_Wet

(d) SPI 24

The tables 6.14(a, b) and 6.15(c, d) are the AIRS2's confusion matrices for test cases SPI3-SPI24 test experiments. The AIRS2's confusion matrices had fewer zeroes along the main diagonal in comparison to the MLP and NaiveBayes algorithms. The may have been fewer zeroes on the main diagonal, but the values were very low, making us draw a similar conclusion as that of the MLP and NaiveBayes algorithm. The values in the first columns of the confusion matrices are indicative that the AIRS2 was also biased towards predicting most instances as 'Near_Normal'. This helps us explain the poor performance measures of the AIRS2 algorithm as shown by the bar chart of figures 6.2-6.9.

6.7. Discussion

Saeid et al [63] performed an investigation where they used EDI and SPI on a number of ANN algorithms. The algorithms were tested on several rainfall stations in the Tehran Province of Iran. In their experiments they use time scales 3, 6, 9 and 12 for SPI. In their test results it was found that the ANN had a 70% to 85% correct classification of SPI with lead time 6 months. For SPI with lead time of 9 months, the ANN had correct classification of 59% to 80% for all test cases. For SPI with lead time 12 months, the ANN had correct classification that ranged between 54-62%. The results revealed by Saed et al [63] are slightly similar to those MLP in the drought case study.

For the NaiveBayes algorithms, Liu et al [51] ran experiments on a set of meteorological data collected from the Hong Kong Observatory Headquarters. In their experiments they found that the NaiveBayes classifier had an overall classification performance ranging between 65.22% and 88.77% for all the test cases. In their experiments they did not use SPI as they used several attributes like wind, speed, temperature and humidity (just to mention a few). The results they revealed were slightly better than those revealed by the Trompsburg case study of this dissertation. This may be due to choice of parameters for classifying rainfall and precipitation.

Peng et al. [67] tested the AIRS2 algorithm on four famous datasets, the Iris, Ionosphere, Diabetes and Sonar datasets. In their experiments it was found that the AIRS2 had a classification accuracy ranging from 74.2%-96.0% across all test cases. The authors also noted that when the AIRS2 algorithm's performance is compared to some traditional methods, the AIRS2 had lower classification accuracy. These are similar trends in performance that were experienced in the case studies presented in the dissertation. Throughout all the case studies the performance of the AIRS2 is not as good as that of the MLP and the NaiveBayes.

6.8. Conclusion

The statistical experiments conducted above for algorithm performance measures indicate that the mean Kappa statistic for the MLP, NaiveBayes and AIRS2 algorithms were statistically similar. The mean F-measure for the MLP and NaiveBayes were also statistically similar.

Overall, across all performance measures, one can then safely conclude that there was a significant difference in mean algorithm performance measures for the MLP, NaiveBayes and AIRS2 algorithms.

When the bar charts in Figures 6.2 to 6.9 are carefully examined, one finds that the three algorithms had a mediocre to poor classification performance. The average rate for most performance measures was below 65% and in some cases, 15%. This is indicative that the three algorithms will produce mediocre classification results if used on a drought case study.

7. CONCLUSION

The rapid development of computing hardware and software has made capturing and storing data fairly easy and inexpensive [62]. When monitoring activities from domains such as network management, drought & climate observations, and crisis management, one is faced with large numbers of devices producing and collecting data from various sources of information. The available and inexpensive means of data storage have made possible the collection in real-time of large volumes of data. This has raised the need for data mining techniques to perform live performance monitoring and analysis of data. Building upon the emergence of soft computing methods and techniques in data mining, this dissertation has identified and applied three of the most known soft computing techniques, viz. Artificial Immune Systems (AIS), Bayesian Networks (BN) and Artificial Neural Networks (ANN) to three scenarios: Wi-Fi network monitoring, Network Intrusion detection and Drought Monitoring. This was with the aim of investigating how the three methods perform for the three datasets, and finding which of the three methods are better suited for use in each of the case study scenarios.

In the RedButton Wi-Fi network monitoring case study, for anomaly and aberrant behaviour in network performance detection, it was found that there is a significant statistical difference in performance among the three soft computing techniques. The BN was able to identify more detectors than the ANN, but ANN performed detection with much higher correctness in classification, accuracy and precision. The AIS displayed poor performance for anomaly detection in network monitoring.

Intrusion was then introduced into the RedButton's network dataset, and the computing techniques were again assessed. This formed part of the second cases study for the dissertation. The assessment revealed as compared to the case without intrusion, the performance for the soft computing techniques have decreased after intrusion was introduced in the network. The results revealed a highly significant statistical difference in performance for the three techniques. The ANN performed better than the BN and the AIS. Its performance was higher in term of it correctness in classification, accuracy and precision. This reveals the effect of intrusion on the soft computing techniques, as they slightly declined in accuracy of classification.

The third case study of the dissertation was on drought monitoring in Trompsburg in the Free State in South Africa. This was conducted in the effort to find out the difference in algorithm performance and which of the three methods are better suited for drought monitoring. The results from the investigation revealed that there was not enough statistical difference in performance for the Kappa statistic performance of the algorithms; and again there was statistical similarity in F-measure algorithm performance among the ANN and BN techniques. Overall, when the performance of the three algorithms was assessed, one found that the overall performance was not satisfactory, with an average correctness and accuracy of just above 60%. In terms of computation performance and accuracy, 60% can be classified as poor performance.

The performance of the algorithms tested in this thesis can be partially explained by the effect that the dataset size and feature selection technique has on the performance of an algorithm. This was investigated by Catal et al in [9]. In their experiments they tested nine classifier algorithms on five publicly available NASA datasets for software fault prediction.

The NaiveBayes and the MLP algorithms were amongst the nine algorithms that were tested in their experiments. In their results it was found that the NaiveBayes does not perform as well as the Random Forest Classifier on large datasets, but it produces better accuracy performance when applied on small datasets. It is well-known that machine learning algorithms perform better when datasets are large, but this was not the case for the NaiveBayes in these experiments.

For large datasets, it was found that the AIRS2 algorithm had lower classification accuracy, and these are similar results that were revealed in the three case study experiments.

Future Work

Future direction of the work may include:

- Drought monitoring: The soft computing techniques used for experimentation in this dissertation proved to be inefficient in drought monitoring. They revealed poor performance. This call for a further investigation into finding soft computing techniques that can be used for drought monitoring.
- Pollution monitoring: There is an interest in the climate change, and there have been sensors like the Waspmote XBee, with Gas Sensor board extension and GPS location module. The data collected from these sensors can be valuable in climate change monitoring and reporting.

8. BIBLIOGRAPHY

[1] AB D UL-KADER, H.M. 2010. Artificial immune clonal selection classification algorithms for classifying malware and benign processes using API call sequences. *International Journal of Computer Science and Network Security* 10, 31. .

[2] ABRAMSON, B., BROWN, J., EDWARDS, W., MURPHY, A. AND WINKLER, R.L. 1996. Hailfinder: A Bayesian system for forecasting severe weather. *International Journal of Forecasting* 12, 57-71. .

[3] ANTONIĆ, O., KRIŽAN, J., MARKI, A. AND BUKOVEC, D. 2001. Spatio-temporal interpolation of climatic variables over large region of complex terrain using neural networks. *Ecological Modelling* 138, 255-263. .

[4] ANTONIO, S.C., RAFAEL, C., CARMEN, S. AND JOSE, M.G. 2002. Bayesian Networks For Probalistic Weather Prediction. In Anonymous , 695-700.

[5] BROOKS JR, F.P. 1996. The computer scientist as toolsmith II. *Communications of the ACM* 39, 61. .

[6] BUSTAMI, R., BESSAIH, N., BONG, C. AND SUHAILI, S. 2007. Artificial neural network for precipitation and water level predictions of bedup river. *IAENG International Journal of Computer Science* 34, .

[7] CANNADY, J. 1998. Artificial neural networks for misuse detection. In Arlington, VA, USA, 5–8 October 1998, Anonymous , 368-381.

[8] CANNADY, J. 2000. Applying CMAC-Based On-Line Learning to Intrusion Detection. In Como, Italy, 24–27 July 2000, Anonymous IEEE Press, , 405-410.

[9] CATAL, C. AND DIRI, B. 2009. Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. *Information Sciences* 179, 1040-1058. .

[10] CHARNIAK, E. 1991. Bayesian Networks Without Pain. 12 Number 4, 50-63. .

[11] CHENG, E., JIN, H., HAN, Z., SUN, J., LU, X. AND ZHAO, W. 2005. Network-Based Anomaly Detection Using an Elman Network. In *Networking and Mobile Computing, Lecture Notes in Computer Science,* Anonymous Springer Berlin / Heidelberg, , 471-480.

[12] COLEMAN, D. AND DIENER, N. 2007. Protecting WiFi Networks from Layer 1 Security Threats. .

[13] DASGUPTA, D. AND AND SAHA, S. 2009. A biologically inspired password authentication system. In *5th Annual Workshop on Cyber Security and information intelligence Research: Cyber Security and information intelligence Challenges and Strategies,* Oak Ridge, Tennessee, April 13 - 15, 2009, F. SHELDON, G. PETERSON, A. KRINGS, R. ABERCROMBIE AND A. MILI, Eds. , 1.

[14] DASGUPTA, S. AND FORREST, S. 1995. Tool Breakage Detection in Milling Operations using a Negative-Selection Algorithm. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.8640.

[15] DASGUPTA, D. AND ATTOH-OKINE, N. 1997. Immunity-based systems: a survey. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, Anonymous , 369-374 vol.1.

[16] DASGUPTA, D. AND GONZALEZ, F. 2002. An immunity-based technique to characterize intrusions in computer networks. *Evolutionary Computation, IEEE Transactions on Evolutionary Computation* 6, 281-291. .

[17] DE CASTRO, L.N. AND VON ZUBEN, F.J. 2000. Artificial Immune Systems: Part II–A Survey of Applications. .

[18] DIETTERICH, T.G. 1998.
Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation* 10, 1895-1923. .

[19] DOUGLAS, A.P., BREIPOHL, A.M., LEE, F.N. AND ADAPA, R. 1998. The impacts of temperature forecast uncertainty on Bayesian load forecasting. *Power Systems, IEEE Transactions on* 13, 1507-1513. .

[20] DUNNE, R.A. 2007. A statistical Approach to Neural Network for Pattern Recognition. Wiley-Interscience, .

[21] ERMAN, A.T., HOESEL, L.V., HAVINGA, P. AND WU, J. 2008. Enabling mobility in heterogeneous wireless sensor networks cooperating with UAVs for mission-critical management. *Wireless Communications, IEEE* 15, 38-46. .

[22] FORBES, J., HUANG, T., KANAZAWA, K. AND RUSSELL, S. 1995. The BATmobile: towards a Bayesian automated taxi. In Anonymous , 1878-1885.

[23] FORREST, S., PERELSON, A.S., ALLEN, L. AND CHERUKURI, R. 1994. Self-nonself discrimination in a computer. In *Research in Security and Privacy, 1994. Proceedings., 1994 IEEE Computer Society Symposium on*, Anonymous , 202-212.

[24] FORREST, S., HOFMEYR, S.A., SOMAYAJI, A. AND LONGSTAFF, T.A. 1996. A sense of self for Unix processes. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, Anonymous , 120-128.

[25] FRANK, E., HALL, M. AND TRIGG, L. Weka 3: Data Mining Software in Java. 3, http://www.cs.waikato.ac.nz/ml/weka/index.html.

[26] FRIENDLY, H. Wireless Netwoking in the Developing World. .

[27] GARTNER INC. 2004. The Number of Hot Spot Users Worldwide to Triple in 2004. 2011, .

[28] GARTNER INC. 2011. 428 Million Mobile Communication Devices Sold Worldwide in First Quarter 2011, a 19 Percent Increase Year-on-Year. 2011, .

[29] GERHARDS, R. 2009. The Syslog Protocol - RFC 5424. 2010, .

[30] GHOSH, A.K. AND SCHWARTZBARD, A. 1999. A study in using neural networks for anomaly and misuse detection. In Berkeley, CA, USA, August 23-26, 1999, Anonymous USENIX Association, , 12-12.

[31] GHOSH, A., MICHAEL, C. AND SCHATZ, M. 2000. A Real-Time Intrusion Detection System Based on Learning Program Behavior. 1907, 93-109. .

[32] GUTTMAN, N.B. 1998. Comparing the palmer drought index and the standardized precipitaion index. *Journal of the American Water Resources Association* 34, 113. .

[33] HAGAN, M.T., DEMUTH, H.B. AND BEALE, M.H. 1995. Neural Network Design. PWS Publication Company; Har/Dis edition, .

[34] HALL, J. 2004. Enhancing intrusion detection in wireless networks using radio frequency fingerprinting. In *In Proceedings of the 3rd IASTED International Conference on Communications, Internet and Information Technology (CIIT,* Anonymous Citeseer, .

[35] HAYES, M.J. 1999. Monitoring the 1996 drought using the standardized precipitation index. *Bulletin of the American Meteorological Society* 80, 429. .

[36] HAYES, M., SVOBODA, M., WALL, N. AND WIDHALM, M. 2011. The Lincoln declaration on drought indices: universal meteorological drought index recommended. *Bulletin of the American Meteorological Society* 92, 485. .

[37] HECKERMAN, D. 1996. A Tutorial on learning With Bayesian Networks. MSR-TR-95-06, .

[38] HOFMANN, A., SCHMITZ, C. AND SICK, B. 2003. Intrusion Detection in Computer Networks with Neural and Fuzzy Classifiers. 2714, 174-174. .

[39] HONGJUN LU, SETIONO, R. AND HUAN LIU. 1996. Effective data mining using neural networks. *Knowledge and Data Engineering, IEEE Transactions on* 8, 957-961. .

[40] HUGHES, B.L. AND SAUNDERS, M.A. 2002. A drought climatology for Europe. 22, 1571-1592. .

[41] INTERNET WORLD STATISTICS. November 25, 2010. World Internet Usage Statistics News and World Population Stats. 2010, .

[42] JAKOBSON, G., LEWIS, L., MATHEUS, C.J., KOKAR, M.M. AND BUFORD, J. 2005. Overview of situation management at SIMA 2005. In *Military Communications Conference, 2005. MILCOM 2005. IEEE,* Anonymous , 1630-1636 Vol. 3.

[43] JAKOBSON, G., BUFORD, J. AND LEWIS, L. 2007. Situation Management: Basic Concepts and Approaches. 18-33. .

[44] JEANNE KELLY. 2007. Understanding the Immune System - How It Works
. National Institute of Allergy and Infectious Diseases Science Education, United States.

[45] JENSEN, F.V., KJAERULFF, U.B., KRISTIANSEN, B., LANGSETH, H., SKAANNING, C., VOMLEL, J. AND VOMLELOVÁ, M. 2001. The SACSO methodology for troubleshooting complex systems. *Artif. Intell. Eng. Des. Anal. Manuf* 14, 321-333. .

[46] KENNETT, R., KORB, K. AND NICHOLSON, A. 2001. Seabreeze Prediction Using Bayesian Networks. 2035, 148-153. .

[47] KJAERULFF, U.B. AND MADSEN, A.L. 2007. Bayesian Networks and Influence Diagrams: Guide To Construction and Analysis. Springer; 1 edition, .

[48] KNIGHT, T. AND TIMMIS, J. 2003. A Multi-Layered Immune Inspired Approach to Data Mining. In Anonymous , 195-201.

[49] KRUEGEL, C., MUTZ, D., ROBERTSON, W. AND VALEUR, F. 2003. Bayesian event classification for intrusion detection. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual,* Anonymous , 14-23.

[50] KYOUNG-JAE K. 2006. Artificial neural networks with evolutionary instance selection for financial forecasting. In 30 April 2006, Anonymous , 519-526.

[51] LIU, J.N.K., LI, B.N.L. AND DILLON, T.S. 2001. An improved naive Bayesian classifier technique coupled with a novel input solution method [rainfall prediction]. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 31, 249-256. .

[52] LLOYDHUGHES, B. 2002. A drought climatology for Europe. *International Journal of Climatology* 22, 1571. .

[53] LONVICK, C. 2001. The BSD syslog Protocol -RFC 3164. 2010, .

[54] LUTHER, K., BYE, R., ALPCAN, T., MULLER, A. AND ALBAYRAK, S. 2007. A Cooperative AIS Framework for Intrusion Detection. In *Communications, 2007. ICC '07. IEEE International Conference on,* Anonymous , 1409-1416.

[55] MATT MCGEE. 2010. By The Numbers: Twitter vs Facebook vs Google Buzz. 2011, .

[56] MCKEE, T.B., DOESKEN, N.J. AND KLEIST, J. 1993. The relationship of drought frequency and duration to time scales. 8, 17-22. .

[57] MCKEE, T.B., DOESKEN, N.J. AND KLEIST, J. 1995. Drought Monitoring with Multiple Time Scales. 9, 233-236. .

[58] MEHROTRA, K., MOHAN, C.K. AND RANKA, S. 1996. Elements of Neural Network. The MIT Press, .

[59] MICROSOFT CORPORATION. 2012. About statistical analyses tools. 2012, .

[60] MISHRA, A.K. AND DESAI, V.R. 2006. Drought forecasting using feed-forward recursive neural network. *Ecological Modelling* 198, 127-138. .

[61] MISHRA, A.K. AND SINGH, V.P. 2010. A review of drought concepts. *Journal of Hydrology* 391, 202-216. .

[62] MITRA, S., SANKAR, K.P. AND PABITRA, M. 2002. Data mining in soft computing framework: A survey. *IEEE Transactions on Neural Networks* 13, 3-15. .

[63] MORID, S., SMAKHTIN, V. AND BAGHERZADEH, K. 2007. Drought forecasting using artificial neural networks and time series of drought indices. *International Journal of Climatology* 27, 2103-2111. .

[64] NOOR, K.B.M. 2008. Case study: a strategic research methodology. *American journal of applied sciences* 5, 1602. .

[65] PANDA, M. 2007. Network intrusion detection using naive bayes. *International Journal of Computer Science and Network Security* 7, 258. .

[66] PEARL, J. 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, .

[67] PENG, L., PENG, Y., LIU, X., LIU, C., ZENG, J., SUN, F. AND LU, Z. 2007. A Supervised Classifier Based on Artificial Immune System. 4488, 355-362. .

[68] QUIRING, S.M. 2003. An evaluation of agricultural drought indices for the Canadian prairies. *Agricultural and Forest Meteorology* 118, 49. .

[69] RIDDEL, J. 2007. PacketCable Implementation. Cisco Press, .

[70] ROBINSON, S. 2004. Simulation: The Practice of Model Development and Use 1st Edition. John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England.

[71] ROJAS, R. AND FIELDMAN, J. 1996. Neural Networks: A Systematic Introduction. Springer, New York, NY, USA.

[72] ROKACH, L. AND MAIMON, O. 2008.
Data Mining With Decision Trees: Theory And Applications (Series In Machine Perception & Artifical Intelligence). World Scientific Publishing, Singapore.

[73] SAJIKUMAR, N. 1999. A non-linear rainfall-runoff model using an artificial neural network. *Journal of hydrology* 216, 32. .

[74] SCHAFFER, C. 1991. When does overfitting decrease prediction accuracy in induced decision trees and rule sets? In *
Proceedings of the European working session on learning on Machine learning*, Porto, Portugal, Y. KODRATOFF, Ed. Springer-Verlag New York, Inc., New York, NY, USA, 192-205.

[75] SCHINDLER, L.W. 1991. Understanding the Immune System. DIANE Publishing, United States Of America.

[76] SIMS, A.P. 2002. Adopting drought indices for estimating soil moisture: A North Carolina case study. *Geophysical Research Letters* 29, 1183. .

[77] STALLINGS, W. 1993. SNMP, SNMPv and CMIP. .

[78] SUBBULAKSHMI, C.V., DEEPA, S.N. AND MALATHI, N. 2012. Comparative analysis of XLMiner and WEKA for pattern classification. In *Advanced Communication Control and Computing Technologies (ICACCCT), 2012 IEEE International Conference on,* Anonymous , 453-457.

[79] TAN, K. 1995. The application of neural networks to UNIX computer security. *Neural Networks, 1995.Proceedings., IEEE International Conference on* 1, 476. .

[80] TAYLOR, A. 2011. Faminine in East Africa. 2012, .

[81] TAYLOR, D. AND CORNE, D. 2003. An Investigation of the Negative Selection Algorithm for Fault Detection in Refrigeration Systems. In *Lecture Notes in Computer Science: Artificial Immune Systems,* Anonymous Springer, Berlin / Heidelberg, 34-45.

[82] THE ECONOMIST. 2003. Bubble trouble: Is the "Wi-Fi" wireless internet boom about to turn into a bust? 2011, .

[83] TICHY, W.F. 1998. Should computer scientists experiment more? *Computer* 31, 32. .

[84] TYLMAN, W. 2008. Anomaly-Based Intrusion Detection Using Bayesian Networks. In *Dependability of Computer Systems, 2008. DepCos-RELCOMEX '08. Third International Conference on,* Anonymous , 211-218.

[85] TYLMAN, W. 2008. Misuse-Based Intrusion Detection Using Bayesian Networks. In *Dependability of Computer Systems, 2008. DepCos-RELCOMEX '08. Third International Conference on,* Anonymous , 203-210.

[86] UJENEVA, E.L. AND ABIODUN, B.J. 2011. Drought in Western Cape: Statistical Analysis of the Trend and Variability in Present-day and Future Climates. .

[87] VAUGHAN-NICHOLS, S.J. 2003. The Challenge of Wi-Fi Roaming. *IEEE Computer Society* 36, 17-19. .

[88] WATKINS, A., TIMMIS, J. AND BOGGESS, L. 2004. Artificial Immune Recognition System (AIRS): An Immune-Inspired Supervised Learning Algorithm. *Genetic Programming and Evolvable Machines* 5, 291-317. .

[89] WILLIAMS, P., ANCHOR, K., BEBO, J., GUNSCH, G. AND LAMONT, G. 2001. CDIS: Towards a Computer Immune System for Detecting Network Intrusions. 2212, 117-133. .

[90] WINSTON, T. 1997. Introduction to case study. *The Qualitative report* 3, .

[91] WITTEN, I.H. AND FRANK, E. 2005. Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann, .

[92] WU, S.X. AND BANZHAF, W. 2010.
The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing Journal* 10, 1-35. .

[93] XU, C., LI, T., HUANG, X. AND JIANG, Y. 2005. A Weather Forecast System Based on Artificial Immune System. 3611, 426-426. .

[94] YIN, R.K. 2009. Case study research: Design and methods. .

[95] ZHANG, J. AND ZULKERNINE, M. 2006. Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection. In *Communications, 2006. ICC '06. IEEE International Conference on*, Instanbul, 11-15 June, 2006, Anonymous IEEE Xplore, , 2388-2393.