2018

Extraction of patterns in selected network traffic for a precise and efficient intrusion detection approach

Priya Naran Rabadia
Edith Cowan University

Edith Cowan University

Copyright Warning

Extraction of patterns in selected network traffic for a precise and efficient intrusion detection approach

by

Priya Naran Rabadia

This thesis is presented for the degree of

Doctor of Philosophy

School of Science

Edith Cowan University

Perth, Western Australia

August 2018

Abstract

This thesis investigates a precise and efficient pattern-based intrusion detection approach by extracting patterns from sequential adversarial commands. As organisations are further placing assets within the cyber domain, mitigating the potential exposure of these assets is becoming increasingly imperative. Machine learning is the application of learning algorithms to extract knowledge from data to determine patterns between data points and make predictions. Machine learning algorithms have been used to extract patterns from sequences of commands to precisely and efficiently detect adversaries using the Secure Shell (SSH) protocol. Seeing as SSH is one of the most predominant methods of accessing systems it is also a prime target for cyber criminal activities.

For this study, deep packet inspection was applied to data acquired from three medium interaction honeypots emulating the SSH service. Feature selection was used to enhance the performance of the selected machine learning algorithms. A pre-processing procedure was developed to organise the acquired datasets to present the sequences of adversary commands per unique SSH session. The pre-processing phase also included generating a reduced version of each dataset that evenly and coherently represents their respective full dataset. This study focused on whether the machine learning algorithms can extract more precise patterns efficiently extracted from the reduced sequence of commands datasets compared to their respective full datasets. Since a reduced sequence of commands dataset requires less storage space compared to the relative full dataset. Machine learning algorithms selected for this study were the Naïve Bayes, Markov chain, Apriori and Eclat algorithms

The results show the machine learning algorithms applied to the reduced datasets could extract additional patterns that are more precise, compared to their respective full datasets. It was also determined the Naïve Bayes and Markov chain algorithms are more efficient at processing the reduced datasets compared to their respective full datasets. The best performing algorithm was the Markov chain algorithm at extracting more precise patterns efficiently from the reduced datasets. The greatest improvement in processing a reduced dataset was 97.711%. This study has contributed to the domain of pattern-based intrusion detection by providing an approach that can precisely and efficiently detect adversaries utilising SSH communications to gain unauthorised access to a system.

I certify that this thesis does not, to the best of my knowledge and belief:

i. *incorporate without acknowledgment any material previously submitted for a degree or diploma in any institution of higher education;*

ii. *contain any material previously published or written by another person except where due reference is made in the text of this thesis; or*

iii. *contain any defamatory material;*

Signed: 

Date: 30/08/2018

Acknowledgement

Firstly, I would like to thank my family (Dad, Mum, Davina and Parisha) you have wholeheartedly supported and encouraged me throughout this journey. There had been rough times, but your words of encouragement have got me through. This journey would have been a lot harder without you all.

To my partner, Harish, thank you for boosting my spirits when I needed it the most. In addition to listening to my long rants through the tough times and celebrating every small achievement. I would have struggled without you constantly pushing me to finish this journey one day at a time.

Of course, my gratitude to my supervisory panel Craig, Zubair, Andrew and Peter. Your guidance and support had been pivotal to the completion of this journey. Thank you for sharing your knowledge and for the feedback provided. Special thank you to Tony Watson for providing valuable feedback and comments.

To ECU's School of Science and the Security Research Institute (SRI) for giving me the opportunity to undertake this degree. In addition to opportunities that had been presented during this time.

Table of Contents

1 Introduction **1**
1.1 Background of the Study **1**
1.2 Purpose and Scope of Study **4**
1.3 Research Questions and associated Hypotheses **5**
1.4 Thesis Structure **6**

2 Literature Review **7**
2.1 Overview of Intrusion Detection Approaches **7**
2.1.1 Open-source Intrusion Detection Datasets 8
2.1.2 Honeypots 9
2.1.3 Issues in Intrusion Detection 13
2.2 Deep Packet Inspection (DPI) **13**
2.2.1 Adversary Pattern-Based Intrusion Detection 14
2.3 Machine learning **16**
2.3.1 Probability Theory 18
2.3.2 Probabilistic Classification algorithms 20
2.3.3 Association Rule Mining Algorithm 23
2.3.4 Enhancing Intrusion Detection Approaches 26
2.4 Conclusion **28**

3 Research Methodology and Design **30**
3.1 Research Methodology **30**
3.2 Research Questions **33**
3.3 Research Variables **34**
3.3.1 Dependent Variables 34
3.3.2 Independent Variables 34
3.3.3 Controlled Variables 35
3.3.4 Confounding Variables 35
3.4 Research Procedure **35**
3.4.1 Phase One, Project Understanding Phase 37
3.4.2 Phase Two, Data Understanding Phase 39
3.4.3 Phase Three, Pre-Processing Phase 41
3.4.4 Phase Four, Experimental Phase 43
3.4.5 Phase Five, Evaluation and Analysis Phase 45
3.5 Data Analysis **47**
3.6 Equipment and Resources **49**
3.6.1 Equipment 49
3.6.2 Resources 49
3.7 Ethical Considerations **51**

4 Preliminary Analysis **52**
4.1 Data Understanding **52**
4.1.1 Dataset One 52
4.1.2 Dataset Two 54
4.1.3 Dataset Three 55
4.1.4 Summary of the Data Understanding Phase 55
4.2 Pre-Processing **56**
4.2.1 Dataset One 58
4.2.2 Dataset Two 62

4.2.3 Dataset Three 67
4.2.4 Data Wrangling 70
4.3 Summary of the Preliminary Analysis 71

5 Results 73
5.1 Probabilistic Classification Algorithms 73
5.1.1 Naïve Bayes Algorithm 73
5.1.2 Markov Chain Algorithm 84
5.2 Association Rule Mining 95
5.2.1 Apriori algorithm 97
5.2.2 Eclat algorithm 120
5.3 Aggregation of Results 143
5.3.1 Hypothesis One 144
5.3.2 Hypothesis Two 145
5.3.3 Hypothesis Three 146
5.3.4 Hypothesis Four 149
5.3.5 Hypothesis Five 151
5.3.6 Hypothesis Six 152

6 Discussion and Findings 154
6.1 Outcomes of Research Questions 154
6.1.1 RQ1: Is there an increase in the number of patterns extracted from the reduced datasets by the machine learning algorithms, compared to the number of patterns extracted from their respective full datasets? 154
6.1.2 RQ2: Are the extracted patterns from the reduced datasets by the machine learning algorithms overall more precise, compared to those extracted from their relevant full datasets? 155
6.1.3 RQ3: Are the machine learning algorithms more efficient at extracting patterns from the reduced datasets, compared to the processing time of their respective full datasets? 157
6.2 Implication of the Research 160
6.2.1 Deep Packet Inspection (DPI) 160
6.2.2 Enhance Machine Learning for Pattern-Based Intrusion Detection 160
6.3 Critical Review of the Research Process 162

7 Conclusion 164
7.1 Research Overview 164
7.1.1 Problem Space in Precise and Efficient Intrusion Detection 164
7.1.2 Research Methodology and Procedure 165
7.2 Implications and Conclusions of This Study 166
7.2.1 Is there an increase in the number of patterns extracted from the reduced datasets by the machine learning algorithms, compared to the number of patterns extracted from their respective full datasets? 166
7.2.2 Are the extracted patterns from the reduced datasets by the machine learning algorithms overall more precise, compared to those extracted from their relevant full datasets? 166
7.2.3 Are the machine learning algorithms more efficient at extracting patterns from the reduced datasets, compared to the processing time of their respective full datasets? ... 167
7.3 Recommendations and Future Research 168
7.4 Final Thoughts 169

References 170

Table of Tables

Table 1.1, Shows the global forecasted increase in the Exabytes per month of Internet Protocol (IP) traffic sent for the years between 2016 and 2021. Adapted from (Cisco, 2017) 2
Table 2.1, Shows the detection methods and properties of the two common intrusion detection categories (Butun, Morgera, & Sankar, 2014) 7
Table 2.2, Shows the three types of honeypots and their properties including examples of existing SSH honeypots (Rabadia et al., 2017) 10
Table 2.3, Shows the four interpretations of probability. Along with the purpose and epistemology of the four interpretations of probability 19
Table 3.1, Shows the reason for research, ontology, epistemology, mode of inquiry for the four research paradigms. Adapted from Creswell (2014), Mackenzie and Knipe (2006) and Williamson and Johnson (2013) 31
Table 3.2, Shows a confusion matrix used to measure the performance of a classification learning algorithm. 47
Table 3.3, Shows the hardware equipment, specification and a description of the task conducted on the device used in this study 49
Table 3.4, Shows the software, version and a description of the task conducted using the application within this study 50
Table 4.1, Shows the three SSH datasets that will be utilised in this study 52
Table 4.2, Shows the number of samples in the input table of the six Kippo SSH honeypots combined to form SSH dataset one 54
Table 4.3, Shows a sample of the transposed and tidied dataset 60
Table 4.4, Shows a sample of the formatted, transposed and tidied dataset 61
Table 5.1, Shows the precision, accuracy, sensitivity, F1 score and error rate of the classified instances using the trained Naïve Bayes algorithm for RD1 and FD1. Calculated using the confusion matrices for each dataset, the measurements are on a scale between 0 and 1, where 1 is the highest. 76
Table 5.2, Shows the precision, accuracy, sensitivity, F1 score and error rate of the classified instances using the trained Naïve Bayes algorithm for RD2 and FD2. Calculated using the confusion matrices for each dataset, the measurements are on a scale between 0 and 1, where 1 is the highest. 79
Table 5.3, Shows the precision, accuracy, sensitivity, F1 score and error rate of the classified instances using the trained Naïve Bayes algorithm for RD3 and FD3. Calculated using the confusion matrices for each dataset, the measurements are on a scale between 0 and 1, where 1 is the highest. 82
Table 5.4, Shows the mean and standard deviation values for the standard error matrix, lower endpoint matrix and upper endpoint matrix of the transition matrices generated by the Markov chain algorithm for RD1 and FD1. 86
Table 5.5, Shows the mean and standard deviation values for the standard error matrix, lower endpoint matrix and upper endpoint matrix of the transition matrices generated by the Markov chain algorithm for RD2 and FD2. 89
Table 5.6, Shows the mean and standard deviation values for the standard error matrix, lower endpoint matrix and upper endpoint matrix of the transition matrices generated by the Markov chain algorithm for RD3 and FD3. 93
Table 5.7, Shows the results from the initial analysis conducted on the three distinct datasets, in order to determine the minimum support and confidence values to be set along with the length of the rules to be extracted. 98

Table 5.8, Shows the total number of rules extracted, number of unique rules after removing the duplicate rules and the percentage of duplicate rules extracted by the Apriori algorithm applied to RD1 and FD1. 98
Table 5.9, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Apriori algorithm applied to RD1 and FD1. 101
Table 5.10, Shows the total number of rules extracted, number of unique rules after removing the duplicate rules and the percentage of duplicate rules extracted by the Apriori algorithm applied to RD2 and FD2. 106
Table 5.11, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Apriori algorithm applied to RD2 and FD2. 109
Table 5.12, Shows the total number of rules extracted, number of unique rules after removing the duplicate rules and the percentage of duplicate rules extracted by the Apriori algorithm applied to RD3 and FD3. 113
Table 5.13, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Apriori algorithm applied to RD3 and FD3. 116
Table 5.14, Shows the total number of rules extracted, number of unique rules after removing the duplicate rules and the percentage of duplicate rules extracted by the Eclat algorithm applied to RD and FD1. 121
Table 5.15, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Eclat algorithm applied to RD1 and FD1. 125
Table 5.16, Shows the total number of rules extracted, the number of unique rules after removing the duplicate rules and the percentage of duplicate rules extracted by the Eclat algorithm applied to RD2 and FD2. 129
Table 5.17, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Eclat algorithm applied to RD2 and FD2. 132
Table 5.0.18, Shows the total number of rules extracted, number of unique rules after removing the duplicate rules and the percentage of duplicate rules extracted by the Eclat algorithm applied to RD3 and FD3. 136
Table 5.19, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Eclat algorithm applied to RD3 and FD3. 140
Table 5.20, Shows the aggregated True Positive (TP) rate of the trained Naïve Bayes classifier applied to the test sets of the three reduced datasets and their respective full datasets, along with the difference between the rates. 144
Table 5.21, Shows the aggregated degree of freedom for reduced datasets and their respective full datasets calculated by the Markov chain algorithm. 145
Table 5.22, Shows the aggregated unique rule sets for the reduced datasets and their respective full datasets after the duplicated rules had been removed by the Apriori algorithm 145
Table 5.23, Shows the aggregated unique rule sets for the reduced datasets and their respective full datasets after the duplicated rules had been removed by the Eclat algorithm 146
Table 5.24, Shows the aggregated precision of the trained Naïve Bayes classifier applied to the three reduced datasets and their respective full datasets 147
Table 5.25, Shows the mean and standard deviation values for the standard error matrix, lower endpoint matrix and upper endpoint matrix of the transition matrices generated by the Markov chain algorithm for the three reduced datasets and their respective full datasets. 149
Table 5.26, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Apriori algorithm applied to the three reduced datasets and the respective full datasets. 150

Table 5.27, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Eclat algorithm applied to the three reduced datasets and the respective full datasets. 151
Table 5.28, Shows the mean processing time in seconds(s) for the Naïve Bayes, Markov chain, Apriori and Eclat algorithms to process reduced datasets and their respective full datasets iterated 100 times (Test 1) and 1,000 times (Test 2) 153

Table of Figures

Figure 2.1, illustrates the three main components of SSH-2, transport layer protocol, user authentication layer protocol and connection layer protocol *12*
Figure 3.0.1, Illustrates the research procedure designed for this study consisting of five phases. *37*
Figure 3.0.2, Illustrates a flowchart showing the process of phase 1, the project understanding phase *38*
Figure 3.0.3, Illustrates a flowchart showing the process of phase 2, the data understanding phase. *40*
Figure 3.0.4, Illustrates a flowchart showing the process of phase 3, the pre-processing understanding phase. *42*
Figure 3.0.5, Illustrates a flowchart showing the process of phase 4, the experimental phase. *44*
Figure 3.0.6, Illustrates a flowchart showing the process of phase 5, evaluation and analysis phase *46*
Figure 4.0.1, Illustrates the MySQL dataset structure for Kippo honeypot (Rabadia et al., 2017) *53*
Figure 4.0.2, Depicts the five steps in the pre-processing procedure used in this study. The data wrangling step is outside the boundary as the step will occur prior to applying a chosen machine learning algorithm. *57*
Figure 4.0.3, Depicts the three stages of data massaging in the data filtering step of the pre-processing procedure *64*
Figure 5.0.1, Illustrates the processing time in seconds(s) for the Naïve Bayes algorithm to process RD1 and FD1. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. *77*
Figure 5.0.2, Illustrates the processing time in seconds(s) for the Naïve Bayes algorithm to process RD1 and FD1. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. *77*
Figure 5.0.3, Illustrates the processing time in seconds(s) for the Naïve Bayes algorithm to process RD2 and FD2. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. *80*
Figure 5.0.4, Illustrates the processing time in seconds(s) for the Naïve Bayes algorithm to process RD2 and FD2. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. *80*
Figure 5.0.5, Illustrates the processing time in seconds(s) for the Naïve Bayes algorithm to process RD3 and FD3. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. *83*
Figure 5.0.6, Illustrates the processing time in seconds(s) for the Naïve Bayes algorithm to process RD3 and FD3. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. *83*
Figure 5.0.7, Illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for the RD1 and FD1, scaled to log_{10}*. The process was*

iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 87
Figure 5.0.8, Illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for RD1 and FD1, scaled to log_{10}. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 88
Figure 5.0.9, Illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for RD2 and FD2. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 91
Figure 5.0.10, Illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for the RD2 and FD2. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 91
Figure 5.0.11, Illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for RD3 and FD3. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 94
Figure 5.0.12, Illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for RD3 and FD3. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 95
Figure 5.0.13, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Apriori algorithm from FD1. 99
Figure 5.0.14, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Apriori algorithm from RD1. 99
Figure 5.0.15, Illustrates a scatter plot of support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from FD1. 100
Figure 5.0.16, Illustrates a scatter plot of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from RD1. 101
Figure 5.0.17, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from FD1. 102
Figure 5.0.18, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from RD1. 103
Figure 5.0.19, Illustrates the processing time in seconds(s) of the Apriori algorithm process RD1 and FD1. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 104
Figure 5.0.20, Illustrates the processing time in seconds(s) of the Apriori algorithm process RD1 and FD1. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 105
Figure 5.0.21, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Apriori algorithm from FD2. 106
Figure 5.0.22, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Apriori algorithm from RD2. 107
Figure 5.0.23, Illustrates a scatter plot of support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from FD2. 108
Figure 5.0.24, Illustrates a scatter plot of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from RD2. 108
Figure 5.0.25, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from FD2. 109

Figure 5.0.26, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from RD2. 110
Figure 5.0.27, Illustrates the processing time in seconds(s) of the Apriori algorithm process RD2 and FD2. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 111
Figure 5.0.28, Illustrates the processing time in seconds(s) of the Apriori algorithm process the RD2 and FD2. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 112
Figure 5.0.29, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Apriori algorithm from FD3. 113
Figure 5.0.30, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Apriori algorithm from RD3. 114
Figure 5.0.31, Illustrates a scatter plot of support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from FD3. 115
Figure 5.0.32, Illustrates a scatter plot of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from RD3. 115
Figure 5.0.33, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from FD3. 117
Figure 5.0.34, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from RD3. 117
Figure 5.0.35, Illustrates the processing time in seconds(s) of the Apriori algorithm process RD3 and FD3. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 119
Figure 5.0.36, Illustrates the processing time in seconds(s) of the Apriori algorithm process RD3 and FD3. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 119
Figure 5.0.37, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Eclat algorithm from FD1. 122
Figure 5.0.38, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Eclat algorithm from RD1. 122
Figure 5.0.39, Illustrates a scatter plot of support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from FD1. 123
Figure 5.0.40, Illustrates a scatter plot of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from RD1. 124
Figure 5.0.41, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from FD1. 125
Figure 5.0.42, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from RD1. 126
Figure 5.0.43, Illustrates the processing time in seconds(s) of the Eclat algorithm process RD1 and FD1. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 127
Figure 5.0.44, Illustrates the processing time in seconds(s) of the Eclat algorithm process RD1 and FD1. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 128
Figure 5.0.45, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Eclat algorithm from FD2. 129
Figure 5.0.46, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Eclat algorithm from RD2. 130
Figure 5.0.47, Illustrates a scatter plot of support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from FD2. 131

Figure 5.0.48, Illustrates a scatter plot of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from RD2. 131
Figure 5.0.49, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from FD2. 133
Figure 5.0.50, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from RD2. 133
Figure 5.0.51, Illustrates the processing time in seconds(s) of the Eclat algorithm process RD2 and FD2. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 134
Figure 5.0.52, Illustrates the processing time in seconds(s) of the Eclat algorithm process RD2 and FD2. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 135
Figure 5.0.53, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Eclat algorithm from FD3. 137
Figure 5.0.54, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Eclat algorithm from RD3. 138
Figure 5.0.55, Illustrates a scatter plot of support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from FD3. 138
Figure 5.0.56, Illustrates a scatter plot of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from RD3. 139
Figure 5.0.57, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from FD3. 141
Figure 5.0.58, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from RD3. 141
Figure 5.0.59, Illustrates the processing time in seconds(s) of the Eclat algorithm process RD3 and FD3. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 142
Figure 5.0.60, Illustrates the processing time in seconds(s) of the Eclat algorithm process RD3 and FD3. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed. 143

1 Introduction

1.1 Background of the Study

Mitigating the exposure of digital assets is becoming increasingly imperative. Estimates have shown cybercrime costs Australia one billion AUD each year (The Department of the Prime Minister and Cabinet, 2016). A defence in depth cyber security strategy can mitigate the exposure of these digital assets. Intrusion detection approaches are deployed by Intrusion Detection Systems (IDSs), to monitor network traffic and trigger an alert, or an appropriate response when unauthorised activities are detected on a network. The concept of intrusion detection was first introduced in 1980 by Anderson (1980). Traditionally there are two types of intrusion detection approaches that are implemented, signature or misuse based, and anomaly based.

The signature or misuse based detection category involves comparing the signatures of current network traffic to a database of known bad signatures of network traffic. If a reasonable match is found an alert is triggered for possible adversarial activities on the network (Sultana, Chilamkurti, Peng, & Alhadad, 2018). The signature or misuse based detection approaches are known for precisely classifying known bad network traffic. However, unknown or new activities go undetected, as the signatures have not yet been generated for a comparison to take place. Previously unknown or new vulnerabilities exposed by adversarial activities (Bilge & Dumitra, 2012) are colloquially known as zero day attacks.

Anomaly based detection approaches involve generating a baseline of normal network traffic behaviour. The generated baseline is then compared to the current network traffic observed. Any nonconformities to the baseline are considered to be anomalous traffic, and an alert is triggered for possible adversarial activities detected on the network (Sultana *et al.*, 2018). Though, any authorised network traffic that does not conform to the baseline also triggers an alert. This impacts the precision of anomaly based intrusion detection approaches to classify network traffic. Unlike the signature or misuse based detection category, these approaches can identify zero day attacks as anomalous traffic generally results in nonconforming traffic to the baseline.

Precision and efficiency are key performance attributes of an intrusion detection approach. A detection approach needs to be precise at detecting both, known and unknown adversarial activities on a network. Along with precisely detecting adversarial activities on a network, an intrusion detection approach needs to efficiently process the network traffic. Allowing for a timely detection of an adversary on the network prior to protected assets being attacked and subsequently compromised (Masduki & Ramli, 2016).

The efficiency of an intrusion detection approach is significant, as predictions show on a global scale the number of Exabytes (10^{18}) per month of IP traffic sent is set to increase with the passing of each year. Table 1.1 shows there is an annual growth rate of 24% in the Exabytes per month of IP traffic between the years of 2016 until 2021 (Cisco, 2017). Table 1.1 is adapted from Cisco's 2017 white paper, that forecasted the trends and predictions of global internet traffic (Cisco, 2017). The increase in Internet traffic can also be aligned with the increase in other network traffic such as that exhibited on organisational networks. Processing the increase in network traffic and precisely detecting adversarial activities is a major challenge.

Table 1.1, Shows the global forecasted increase in the Exabytes per month of Internet Protocol (IP) traffic sent for the years between 2016 and 2021. Adapted from (Cisco, 2017)

Year	***Exabytes (10^{18}) per Month***
2016	96
2017	122
2018	151
2019	186
2020	228
2021	278

Typically, intrusion detection approaches monitor and analyse network flow data or packet level data. Network flow data consists of monitoring and analysing features associated with the flow of data through the network. These features include the source and destination IP address, along with the source and destination port numbers (Hellemons *et al.*, 2012). Monitoring and analysing packet level data have more processing and storage overheads compared to analysing network flow data because there are more features that can be extracted from packet level data compared to network flow data. Examples of additional features that can be extracted include, protocol type and the number of root connections to name a few. This impacts the processing and storage overheads of packet level data analysis.

Deep Packet Inspection (DPI) is the further examination of packet level data to gain a further insight into network traffic communications and can be used for real-time analysis and off-line analysis. There are many uses of DPI within the cyber security domain, including for intrusion detection purposes. The additional knowledge gained from a DPI can be used for cyber defence optimisation, malicious software detection and user activity monitoring including adversary activity identification (Rabadia, Valli, Ibrahim, & Baig, 2017).

The publicly available (open-source) datasets are commonly utilised for benchmarking and evaluating intrusion detection approaches. Since the features required for a DPI are omitted within the open-source datasets, data acquired from three distinct Secure Shell (SSH) honeypots have been utilised for this study. SSH is used for point to point communication over an insecure network, creating an encrypted

tunnel for remote communications and access (Tatu Ylonen, 2017c). SSH was developed in 1995 by Tatu Ylonen and was a response to a password sniffing attack against Helsinki University of Technology. This first version is referred to as SSH-1. The SSH service is predominately targeted by adversaries to attempt to gain unauthorised remote access to a system.

This study aims to investigate adversarial SSH commands for a pattern-based intrusion detection approach. Seeing as SSH is one of the most predominant methods of remotely accessing a systems compared to Telnet and is also a prime vector for cyber criminal activities. Pattern-based intrusion detection approaches discover and extract patterns within network traffic data for adversarial activity detection. Some of the most common means of implementing pattern-based intrusion detection approaches are utilising machine learning algorithms to analyse network traffic. Machine learning is the application of learning algorithms to extract knowledge from data to determine patterns between data points and make predictions. This study utilises selected machine learning algorithms to extract patterns from adversarial SSH commands. The selected machine learning algorithms are the Naïve Bayes, Markov chain, Apriori and Equivalence Class Transformation (Eclat) algorithms.

There are two main approaches for enhancing the precision and efficiency of pattern-based intrusion detection approaches for the SSH service. These involve implementing the use of hybrid algorithms and feature selection of the dataset. Hybrid algorithms are developed by combining two or more algorithms. The notion of a hybrid algorithm is combining machine learning algorithms that complement each other. Algorithms belonging to different machine learning categories are typically combined, such as clustering and classification algorithms (Soheily-Khah, Marteau, & Béchet, 2018). While feature selection is choosing relevant attributes within a dataset that will allow for additional information to be extracted by classifying the dataset based on the selected features. The pre-processing phase is a critical process when applying machine learning algorithms (Malley, Ramazzotti, & Wu, 2016). In the pre-processing phase, the datasets are processed and prepared to be applied to the selected machine learning algorithms. The feature selection process takes place in the pre-processing phase. The pre-processing phase also includes a data reduction step to enhance the performance of the selected machine learning algorithms. Data reduction involves generating a reduced dataset that evenly and coherently represents the respective full dataset. For this study precision is in reference to correctly identifying patterns in sequential adversarial commands within the given datasets. Furthermore for this study efficiency is in reference to the processing time for the selected machine learning algorithms to extract the patterns from the given datasets.

1.2 Purpose and Scope of Study

The pattern-based intrusion detection approach investigated in this study focuses on adversary command data, as a means of detecting adversaries on the monitored network. In conjunction with efficiently processing the network data to timely detect adversaries. This study aims to contribute to the knowledge domain of pattern-based intrusion detection approaches for the SSH service.

The focus of this study was the use of adversary command patterns for intrusion detection purposes, DPI was required to ascertain the adversary command data. The examination of existing literature has shown there has been research conducted in the domain of adversary based intrusion detection approaches utilising DPI (Pimenta Rodrigues *et al.*, 2017). Although there are gaps in the knowledge domain of DPI, this thesis aims to contribute in terms of pattern identification within a sequence of adversary interaction commands. Together with evaluating whether a reduced sequence of command dataset can be efficiently processed and whether the patterns extracted are more precise compared to the respective full dataset.

A pre-processing procedure that encompasses feature selection and data reduction is deployed for this study and is to be applied to the acquired datasets prior to the application of the chosen machine learning algorithms. The selected machine learning algorithms have been utilised to extract patterns from a sequence of adversarial SSH interaction commands. The selected machine learning algorithms are the Naïve Bayes, Markov chain, Apriori and Equivalence Class Transformation (Eclat) algorithms. The Naïve Bayes and Markov chain algorithms are probabilistic classification algorithms and have been selected as they aligned with the machine learning problem of classification and the Bayesian theorem interpretation of probability. The Apriori and Eclat algorithms are association rule mining algorithms and have been selected as they aligned with the machine learning problem of association and the frequentist theorem interpretation of probability.

1.3 Research Questions and associated Hypotheses

The research questions and the associated hypotheses of this thesis are presented below.

RQ1: Is there an increase in the number of patterns extracted from the reduced datasets by the machine learning algorithms, compared to the number of patterns extracted from their respective full datasets?

H1: More class patterns can be extracted from the reduced sequence of commands datasets, compared to those extracted from their respective full datasets by the selected probabilistic classification algorithms.

H2: More patterns can be extracted from the rule sets of the reduced sequence of commands datasets, compared to those extracted from their respective full datasets by the selected association rule mining algorithms.

RQ2: Are the extracted patterns from the reduced datasets by the machine learning algorithms overall more precise, compared to those extracted from their relevant full datasets?

H3: The class patterns extracted by the probabilistic classification algorithms from the reduced sequence of commands datasets are more precise, compared to those extracted from their respective full datasets.

H4: The patterns extracted from the rule sets by the association rule mining algorithms from the reduced sequence of commands datasets are more precise, compared to those rule sets extracted from their respective full datasets.

RQ3: Are the machine learning algorithms more efficient at extracting patterns from the reduced datasets, compared to the processing time of their respective full datasets?

H5: The probabilistic classification algorithms are more efficient at extracting patterns from the reduced sequence of commands datasets, compared to their respective full datasets.

H6: The association rule mining algorithms are more efficient at extracting patterns from the rule set for the reduced sequence of commands datasets, compared to their respective full datasets.

1.4 Thesis Structure

The structure of the remaining thesis is as follows.

Chapter Two is the Literature Review. This is the exploration of the existing literature surrounding this study in order to identify the gap in the knowledge this study intends to fill. Literature surrounding DPI, for adversarial pattern-based intrusion detection approaches are explored, along with identifying the suitable machine learning algorithms to be utilised for this study.

Chapter Three is the Research Methodology and Design for this study. The research approach that underpins this study is established. The research procedure, encompassing five phases is outlined. Further, an overview of data analysis is presented along with the equipment, and resources utilised for this study.

Chapter Four is the Preliminary analysis for this study. Initial exploration and analysis of the acquired datasets are conducted in this chapter along with applying the pre-processing procedure developed for this study. On the conclusion of this chapter, the three SSH honeypot datasets are processed resulting in three reduced datasets and their respective full datasets.

Chapter Five is the Results Chapter. In this chapter, the results collected from the experiments conducted to test the hypotheses of this study. The experiments encompass applying the selected machine learning algorithms to the reduced datasets and their respective full datasets.

Chapter Six is the Discussion and Findings Chapter. The results obtained from the experiments conducted to test the hypotheses in relation to addressing the research questions are analysed. As well as presenting the implications of this study along with a critical review of this study.

Chapter Seven is the Conclusion. In this chapter, an overview of this study and the contributions this thesis has made to the knowledge domain are outlined. Finally, the recommendations and potential further research areas are outlined.

2 Literature Review

2.1 Overview of Intrusion Detection Approaches

This study investigated whether patterns extracted from sequential adversarial SSH commands can be utilised as a pattern-based intrusion detection approach. A pattern-based intrusion detection approach discovers and extracts patterns from network traffic to detect adversaries on the network. The signature-misuse based and anomaly based detection are the two common intrusion detection categories. The detection methods and properties of these common intrusion detection categories are presented in Table 2.1.

Table 2.1, Shows the detection methods and properties of the two common intrusion detection categories (Butun, Morgera, & Sankar, 2014)

Intrusion detection categories	***Detection method***	***Properties of the categories***
Signature-misuse based detection	The signature-misuse based detection approaches compare the current network traffic signature to a database of known 'bad' network traffic signatures. If a significant match is found, an alert is triggered indicating a possible intruder on the network.	Has a low false alert rate compared to anomaly detection approach category, as known 'bad' network traffic signatures are easily identified. The associated risk is new or zero day attacks go undetected, as the network signatures have not been generated yet. Zero day attacks are previously unknown attacks.
Anomaly based detection	Anomaly based detection approaches compare the current network traffic to a generated baseline of normal traffic exhibited on the network. If the current network traffic does not conform to the baseline, it is considered to be anomalous traffic. An alert is triggered indicating a possible intruder on the network.	Unknown or 'zero day' attacks can be identified, as the attack would cause abnormal traffic to be exhibited on the network. There is a higher false alert rate compared to the signature-misuse based detection approach category. As any legitimate network traffic that does not conform to the baseline would trigger an alert.

Precision and efficiency are key attributes of an intrusion detection approach. An intrusion detection approach needs to precisely detect adversaries on the network. There should be a low rate of misclassified network traffic (false alert rate) and high rate of correctly classified network traffic (true alert rate. An intrusion detection approach also needs to be efficient at detecting adversaries on a network to prevent the exposure of assets in a timely manner (Masduki & Ramli, 2016). For this study precision is in reference to correctly identifying patterns in sequential adversarial commands within the given datasets. Furthermore for this study efficiency is in reference to the processing time for the selected machine learning algorithms to extract the patterns from the given datasets.

An intrusion detection approach either monitors packet level data or network flow data. Network flow data consists of monitoring source IP, destination IP, source port and destination port along with the number of packets sent per flow (Hellemons *et al*., 2012). Whereas, packet level monitoring also monitors the protocol type and the number of root connections to name a few attributes. Consequently, monitoring network flow data has less processing and storage overheads compared to monitoring packet level data. However, additional information can be extracted from packet level data compare to network flow data. Within the literature open-source or publicly available datasets are typically utilised when conducting studies on enhancing intrusion detection approaches.

2.1.1 Open-source Intrusion Detection Datasets

The Defence Advanced Research Projects Agency (DARPA) released an intrusion detection dataset in 1998, known as the DARPA 1998 dataset. The DARPA 1998 dataset was derived from a simulated Air Force base network that encompasses fictitious Internet traffic, malicious scripts, injection attacks and Sun BSM attack data (DARPA, 1998). In the year 1999, the DARPA 1998 intrusion detection dataset was revised to contain supplementary attacks as well as Windows NT attack data and is referred to as the DARPA 99 dataset (DARPA, 1999). However, that same year the Knowledge Discovery in Databases (SIGKDD) Cup 99 dataset was released. The data in the KDD cup 99 dataset is formulated from the DARPA 1999 dataset (SIGKDD, 1999). The KDD Cup 99 dataset was originally used for The Third International KDD and Data Mining Tools Competition. Since then the KDD Cup 99 dataset has become a benchmark dataset in the domain of intrusion detection research (Gharib, Sharafaldin, Lashkari, & Ghorbani, 2016; Tavallaee, Bagheri, Lu, & Ghorbani, 2009).

The KDD Cup 99 dataset became the benchmark dataset, as unlike the DARPA 99 dataset users were not required to convert the data from a raw format to a readable format, as well as the necessary features had already been extracted. The KDD Cup 99 dataset contains 14 additional attacks compared to the DARPA 99 dataset. These additional 14 attacks are only seen in the test dataset. Allowing for a detection approach to be evaluated based on unseen attacks, similar to ‘zero day’ attacks that are seen in real network traffic (SIGKDD, 1999; Tavallaee *et al*., 2009). Consequently, there are inherent issues with redundant data within the KDD Cup 99 dataset since it is derived from the DARPA 99 dataset, resulting in classification bias. 78.05% of the KDD Cup 99 training dataset is redundant data and 75.15% of the test dataset is redundant (Tavallaee *et al*., 2009).

In 2009, Tavallaee *et al*. (2009) presented a revised version of the KDD Cup 99 dataset, known as the NSL-KDD dataset. The NSL-KDD dataset addresses some of the issues associated with the KDD Cup 99 dataset. Redundant records had been removed to avoid classification bias. Additionally, allowing data to be randomly selected from the training and testing sets. The data chosen to be included in the

NSL-KDD dataset, have been chosen based on different levels of difficulties to reflect the original KDD Cup 99 dataset for different machine learning algorithms to be tested. Both the KDD Cup 99 dataset and the NSL-KDD dataset have four categories of attacks, Denial of Service (Koch, Golling, & Rodosek), Probe, Remote to Local (R2L) and User to Root (U2R). Additionally, both datasets have 41 features with varying data types between nominal, continuous and binary data types. However, none of the 41 features within the KDD Cup 99 dataset nor the NSL-KDD dataset contain adversary command data. The datasets have issues with high levels of redundant data as well as the absence of adversary command data. From exploring the DARPA 98, DARPA 99, KDD Cup 99 and NSL-KDD datasets are not suitable to be utilised for this study. It has been ~20 years since the benchmark KDD Cup 99 dataset was released and ~10 years since the improved NSL-KDD dataset was released. The focus of this study was to extract patterns from adversarial command interaction data, in reference to the commands utilised by adversaries while interacting with a system.

Other intrusion detection datasets include the PREDICT dataset, UNSW-NB15 dataset and KYOTO honeypot dataset (Gharib *et al*., 2016). The PREDICT dataset was developed by the Department of Homeland Security (DHS) and the Centre of Applied Internet Data Analysis (CAIDA) released to the public in 2016. Access to the dataset is restricted to researchers based in the United States and approved DHS countries. Although Australia is among the list of approved countries, the dataset was not suitable for this study due to risks of data exposure and publication restrictions (CAIDA, 2016; Impact Cyber Trust, 2017). The UNSW-NB15 dataset was released to the public in 2015, it was developed by academics at the University of New South Wales the dataset consists of 49 features and nine attack types (Moustaf & Slay, 2016; Moustafa & Slay, 2015) . Adversary command data is absent deeming the dataset unsuitable for this study. The Kyoto honeypot dataset, was generated by the Kyoto University between 2009 and 2015. The data is collected from Nepenthes low interaction honeypots. The Kyoto dataset contains a total of 24 features (Jungsuk Song, Hiroki Takakura, & Okabe; Kyoto University, 2015). Similar to the previous intrusion detection datasets, the Kyoto honeypot dataset does not contain adversary command data thus deeming the dataset unsuitable for this study.

From evaluating the possible open-source datasets, none of the datasets explored contains adversarial command data. The adversarial command data required for this study was gathered by honeypots. Honeypots are Intrusion Detection Systems (IDSs) (SURF cert IDS, 2013) tools used to gather data on adversary activities.

2.1.2 Honeypots

Honeypots are decoy systems intended to be attacked with the purpose of gathering data of adversarial interactions with the system. These systems are predominantly located in the Demilitarised Zone

(DMZ) of a network, to separate the decoy system from the real network (The Honeynet Project, 2011). There are two categories of honeypots that can be deployed depending on the purpose: production honeypots and research honeypots. Production honeypots are commonly utilised by organisations to gather data on adversaries attempting to gain access to their infrastructure. The data collected from the production honeypots can be used by an organisation to determine potential vulnerabilities in their cyber security strategy that could be exploited by adversaries. This allows the organisation to address the vulnerabilities reducing the probability of adversaries gaining unauthorised access to the network. While data collected from the research honeypots are used to analyse the techniques and methods adversaries use to gain unauthorised access. Data collected from research honeypots can be used to assist organisations in improving their intrusion detection policies and rules. There are three types of honeypots that can be implemented depending on the requirements of the deployment.

The types of honeypots are based on the level of interaction that occurs between the system and the adversary. In Table 2.2, the properties of the different types honeypots and examples of existing Secure Shell (SSH) honeypots are presented. The three types of honeypots are, high interaction, medium interaction and low interaction, honeypots.

Table 2.2, Shows the three types of honeypots and their properties including examples of existing SSH honeypots (Rabadia et al., 2017)

Honeypot type	***Properties***	***Example***
High interaction	These honeypots emulate a fully functioning system. Adversaries are led to believe they are interacting with a real system. The high interaction honeypots have a similar configuration process to a real system, consequentially the maintenance requirements are demanding.	• HonSSH
Medium interaction	These honeypots exhibit selected functionalities of a real system, for example, a SSH session. As a result, the configuration process is simpler than a high-interaction honeypot but the maintenance required is more demanding than a low-interaction honeypot.	• Kippo SSH • Cowrie SSH
Low interaction	These hosts have minimal functionalities compared to a real system. They are simple to configure with low maintenance requirements. Adversaries have minimal interactions with the host compared with a high interaction honeypot.	• Honeyd

The SSH protocol is the selected network traffic chosen to be investigated. Seeing as SSH is one of the most predominant methods of remotely accessing a systems compared to Telnet and is also a prime vector for cyber criminal activities. Since the focus of this study is on a selected network traffic, a medium interaction honeypot was appropriate for this study. A medium interaction honeypot has more

functionalities that are similar to a real system compared to a low interaction honeypot. There are also fewer maintenance requirements compared to a high interaction honeypot. Research based honeypots had been chosen as the data gathered was for academic in this study.

The data acquired for this study had been collected from the Kippo and Cowrie medium interaction SSH honeypots. Kippo SSH honeypots had been adapted from Kojoney honeypots, a low interaction SSH honeypot. The Kippo SSH honeypot was released in 2009 and actively developed until October 2016 (Desaster, 2016). Cowrie developer Michel Oosterhof was a former contributor to the Kippo SSH honeypot project. But after no longer actively being developed he launched and continues to develop the Cowrie SSH honeypot.

In both honeypots, a fake file system can be added and removed, adversaries can interact with a fake file system, reply to attacks and the files downloaded by an adversary are saved for later inspection. The Kippo and Cowrie SSH honeypots are both written in *Python 2.7* and use the Twisted library to emulate a SSH session (Michel Oosterhof, 2018a). The honeypots log forced attempts to gain unauthorised access to the systems as well as the shell session interactions between the honeypot and the adversary.

2.1.2.1 Secure Shell (SSH) Protocol

SSH is used for point to point communications over an insecure network essentially creating an encrypted tunnel for remote communication (Tatu Ylonen, 2017c). SSH was developed in 1995 by Tatu Ylonen and was a response to a password sniffing attack against Helsinki University of Technology and is referred to as SSH-1. The SSH protocol was initially introduced to replace the Telnet protocol due to the lack of encryption. SSH File Transfer Protocol (SFTP) (Tatu Ylonen, 2017b) runs over the SSH protocol and was introduced to replace the insecure File Transfer Protocol (FTP) (Tatu Ylonen, 2017a).

By default, SSH runs on the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) on port 22. The protocol was taken to a working group at the Internet Engineering Task Force (IETF) for standardisation and evolved into SSH-2. In 2006, SSH-2 was assigned a Request For Comment (Network Working Group, 2006b) number and is currently the widely used version of SSH. Figure 2.1, illustrates the three main components of SSH-2, transport layer protocol, user authentication layer protocol and connection layer protocol.

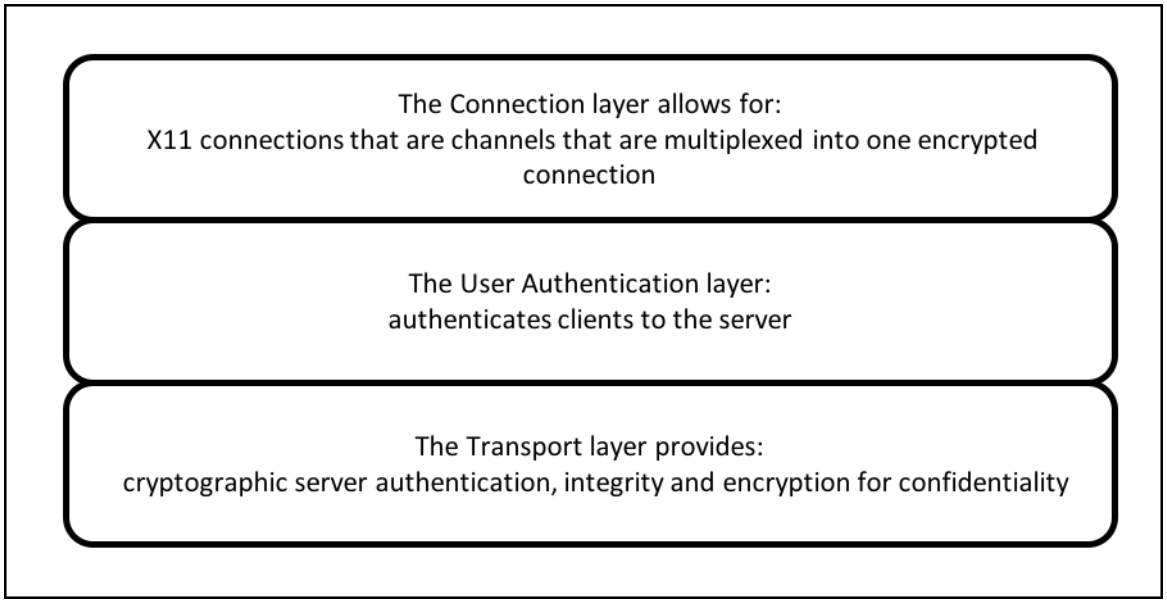


Figure 2.1, illustrates the three main components of SSH-2, transport layer protocol, user authentication layer protocol and connection layer protocol

The transport layer protocol predominantly runs over the Transmission Control Protocol/Internet Protocol (TCP/IP), typically running on port 22. The Transport layer provides cryptographic server authentication, integrity and encryption for confidentiality. All packets sent from one direction should be considered as one stream. Data from both streams should be encrypted using the same encryption algorithm. The 3dec-cbc encryption algorithm is required for encryption if the key lengths are of 112 bits. The integrity of the streams is ensured by the Message Authentication Code (Mackenzie & Knipe) algorithm. The MAC algorithm is a combination of a shared secret (a message the client can verify), packet sequence number and the payload. The required MAC algorithm is hmac-shal, a key length of 20 octets (160 bits). All other layers of the SSH protocol run over the transport layer protocol (Network Working Group, 2006b).

The user authentication layer protocol runs over the Open System Interconnection (OSI) Transport layer protocol, authenticating clients to the server. There are four methods a client can use to authenticate to a server; public key (required), password (optional), host base (optional) and none (not recommended). The common form of authentication is public key and password based. The public key method of authentication uses a pair of cryptographic keys, public and private. The public key of the server is available to everyone and is used for encryption and can only be decrypted by the associated private key. Larger organisations commonly use the password method of authentication as well. The passwords must be encoded with ISO-1064 UFT-8 before transmission by the Transport layer, as stated in RFC 4252. The authentication protocol assumes the Transport layer provides integrity and confidentiality (Network Working Group, 2006a).

The connection layer protocol runs over the Authentication and Transport protocol. The connection layer protocol allows for login sessions, interactive terminal sessions, remote command execution, forwarded TCP/IP and X11 connections are channels that are multiplexed into one encrypted connection. The Connection layer protocol allows for flow control before channels have received a message to indicate a window space is available. The window is the number of bytes that can be sent within the channel. Windows can be adjusted for a larger flow of data however, the window cannot exceed `2^32 − 1` bytes in size. Data can be transferred in both directions, between the client and the server (Network Working Group, 2006c).

2.1.3 Issues in Intrusion Detection

Traditionally intrusion detection approaches fit into the signature-misuse based or anomaly based category. Ideally, an intrusion detection approach needs to be precise and efficient at timely detecting adversaries on a network. This study investigated whether patterns extracted from adversarial SSH commands can be utilised as a pattern-based intrusion detection approach.

The open-source or publicly available intrusion detection databases have been deemed unsuitable for this study. The benchmark KDD Cup 99 dataset has issues with redundant records resulting in classification bias as well as the dataset dating back to 1999. The NSL-KDD dataset was released in 2009 to replace the aging KDD Cup 99 dataset. The data within the NSL-KDD dataset is a subset of the KDD Cup 99 dataset, has inherited some of the issues mentioned. In addition to the absence of adversarial command data within other open-source datasets, adversary command data is also absent. Instead, data for this study was acquired from honeypots, a type of IDS used to gather adversarial data. Medium interaction research honeypots had been chosen that emulate the SSH protocol. Since the focus of this study was the use of adversary command patterns for intrusion detection purposes, Deep Packet Inspection (DPI) techniques have been used to ascertain the adversary command data.

2.2 Deep Packet Inspection (DPI)

DPI is the further examination of packet level data to gain a further insight into network traffic communications and can be used for real-time analysis and off-line analysis. There are many uses of DPI in the cyber security domain, including intrusion detection. The additional knowledge gained from a DPI can be used for cyber defence optimisation, malicious software detection and user activity monitoring including adversary activity identification. Packets from all layers of the TCP/IP can be inspected, including the Application layer of the TCP/IP protocol in which the SSH protocol operates (Pimenta Rodrigues *et al*., 2017).

There are two main challenges of DPI in the context of conducting a forensic investigation (Pimenta Rodrigues *et al*., 2017). Encryption of packet level data can inhibit the readability of the packet and needs to be decrypted before a DPI process can take place. Additionally, deception techniques such as The Onion Router (Keith Stouffer, Victoria Pillitteri, Suzanne Lightman, Marshall Abrams, & Hahn, 2015) are used to maintain the anonymity of the user, as such DPIs can result in incorrect outcomes.

There are also two main intrusion detection challenges of DPI (Pimenta Rodrigues *et al*., 2017). The performance of DPIs can be affected by the volume of data needing to be processed and stored. In addition to the challenge of precisely detecting unauthorised activities, as identifying zero-day attacks would require further investigations to be conducted and the established signatures would need to be updated regularly.

This study focuses on precisely and efficiently detecting adversarial activities for intrusion detection purposes using DPI to extract sequential adversarial commands. The following section examines the literature on adversary command pattern identification for intrusion detection purposes.

2.2.1 Adversary Pattern-Based Intrusion Detection

Studies have been conducted on the use of adversarial command based detection as an intrusion detection approach. Adversarial commands based intrusion detection methods are in reference to, detecting intruder activities based on the commands utilised by an adversary to gain unauthorised access to an asset. There is existing literature with the aim of improving the efficiency of the pattern-based detection within DPI for cyber security purposes.

In the study by Bando, Artan, and Chao (2012), introduced a technique using DPI to optimise regular expressions (RegEx). The authors referred to as Lookahead Finite Automata (LaFA) detection. The aim of the LaFA approach was to address the scalability and redundancy issues of RegEx when processing network data. It was achieved by reducing the data structure of the RegEx string. The proposed LaFA detection by Bando *et al*. (2012) had been shown to need less storage memory compared to the RegEx detection tested. In the current study a reduced sequence of commands dataset requires less storage space compared to the relative full dataset.

Tsai *et al*. (2017) also conducted a study with the aim of improving the DPI process using RegEx strings. The study sought to implement a wildcard pattern search as referred to by the authors, on a Ternary Content Addressable Memory (TCAM) search technique. The wildcard pattern search is based on repeated character sets identified by a finite state machine. Thereafter the TCAM search technique was used to efficiently search for the wildcard patterns. The results showed the TCAM wildcard pattern

search was less memory intensive and more efficient compared to the TCAM RegEx string search. The current study also investigates whether more precise patterns can be extracted from the reduced datasets.

The studies conducted by Bando *et al*. (2012) and Tsai *et al*. (2017) focused on improving the efficiency of a DPI. Whilst, this thesis aimed to improve the precision and efficiency of a DPI. Similar to Bando *et al*. (2012), a reduced dataset is utilised in the procedure to improve the efficiency of the pattern-based intrusion detection approaches. Though unlike the aforementioned studies, this thesis investigated improving the precision and the efficiency of processing a reduced dataset compared to the respective full dataset.

The study conducted by Ramsbrock, Berthier, and Cukier (2007) attempted to profile adversaries based on data collected from a honeypot emulating the SSH service. The data collected was of the usernames and passwords used to attempt to gain access to the system, the number of logins per day and the ratio of failed and successful login attempts data. The study had determined there are seven categories of an attacker's behaviour once they have gained unauthorised access to a system. These categories are checking the software configuration, installing programs, downloading files, running malicious programs, changing account passwords, checking the hardware configuration and changing the system configuration.

The seven categories defined by Ramsbrock *et al.* (2007), had been utilised by the study conducted by Koch et al. (2014). Their study presented a behaviour based detection architecture for intrusion detection in an encrypted environment. The seven categories have been used to develop an attack tree, based on data collected from SSH honeypots and export knowledge. The attack tree developed depicts the probability of a sequence of adversary commands occurring. The results show the attack tree can detect 93.75% of sequential malicious activities compared to 84.70% of single malicious activities. Koch *et al.* (2014) showed an attack tree can be utilised for intrusion detection purposes, however for this study an attack tree was not considered due to the issue of state explosion. State explosion is when the number the states grow exponentially. A full study on the proposed attack tree architecture is yet to be conducted. This thesis aimed to fill the gap in command based detection as an intrusion detection approach, through identifying patterns within adversary command sequences. Unlike the proposed attack trees by Koch *et al*. (2014), this study utilises machine learning algorithms to identified patterns within adversary command sequences.

Research has been conducted on investigating command based detection as an intrusion detection approach. Moon, Pan, and Kim (2016) proposed a malware detection approach that analysed information gathered from behaviour while running processes on a host. Process related features had been extracted from normal and abnormal traffic and referred to as "characteristic parameters". A total

of 39 “characteristic parameters” had been identified. These characteristics are grouped into seven categories. These categories are, process, thread, file system, registry, network, services and miscellaneous. The characteristics are used to determine abnormal and normal behaviour. The C4.5 algorithm is a decision tree algorithm and was used to classify the behaviour exhibited on the network, with a false negative rate of 2% and a false positive rate of 5.8% of abnormal or malware related behaviour detected on a host. The study conducted by Moon *et al*. (2016) provides evidence that process related features can be utilised to determine abnormal or normal behaviour on a host. This provides evidence that command based features could be utilised for the purpose of an intrusion detection.

Kudłacik, Porwik, and Wesołowski (2016) proposed a profile based detection approach for classifying normal and abnormal traffic. Two profiles had been created, a local profile containing data from current users and a fuzzy profile containing generic behaviour. The study used the Schonlau dataset, an open-source UNIX command dataset containing files of normal user commands and adversary commands. The study by Kudłacik *et al*. (2016) focused on the commands of the adversary instead of the sequence of commands as this research intended to investigate. Additionally, this research is focused on investigating the sequence of commands utilised by an adversary on SSH services instead of detecting adversary activity in normal network activities. Studies have focused on using adversary commands to classify normal and abnormal traffic (Caselli, Zambon, & Kargl, 2015). However, there is a gap in knowledge in investigating the sequence of commands an adversary utilises to gain unauthorised access to a system.

The studies examined in this chapter show, there has been research conducted in the area of adversary based intrusion detection approaches utilising DPI. Although there are gaps in the knowledge that was identified this thesis aims to contribute to. This domain of knowledge is within the domain of pattern identification in a sequence of adversary interaction commands. Machine learning algorithms can be used to extract patterns from a sequence of adversary commands.

2.3 Machine learning

Machine learning is the application of learning algorithms to extract knowledge from data to determine patterns between data points and make predictions. A machine learning algorithm can be categorised into three classes: supervised learning, unsupervised learning and semi-supervised learning (Mohri, Rostamizadeh, & Talwalkar, 2014). The algorithms within the supervised learning category are commonly utilised for making predictions. Predictions are formulated by applying the supervised learning algorithms to “pre-labelled” data. The “pre-labelled” data is used to train the algorithms with adjustments made until an acceptable outcome is achieved. Whereas, unsupervised learning algorithms are commonly utilised to determine patterns between data points. The patterns are determined from

unlabelled data and the training process that exists with supervised learning algorithms is omitted. Unsupervised learning algorithms formulate patterns unaided. The third machine learning category is the semi-supervised learning algorithms. Algorithms within this category use procedures from both supervised and unsupervised learning to extract knowledge from a dataset. To evaluate whether a machine learning algorithm is suitable to assist with an enquiry, the four main problem domains machine learning algorithms solve should be examined (Mohri et al., 2014). The four problem domains of machine learning algorithms are, classification, regression, clustering and association (Hastie, Tibshirani, & Friedman, 2013).

The problems solved by supervised learning algorithms are classification and regression issues. The classification problem is solved by training a classifier to form classes based on labelled data. Trained classifiers can be utilised to predict occurrences. The classification solution has the desired outcome of classifying the data into groups. The regression problems also use labelled data to train the predictor. The regression solution is commonly used to predict particular instances based on the existing data to estimate whether there is a dependant or independent relationship between one or more variables.

Clustering and association are the problems solved by unsupervised learning algorithms. The clustering problem is organising data into subsets to form clusters based on observations seen between data points. The outcome of a clustering solution is groups of similar data points, clustering can be utilised for identifying commonality or abnormality in a dataset.

The association problem is identifying association patterns between objects within a dataset (Alpaydin & Bach, 2014). The association solution is discovering patterns between frequently occurring objects in a dataset and calculating the probability of these objects occurring together based on the given data.

The machine learning problems of regression and clustering do not align with the enquiry of this study. The regression problem is used to predict particular values occurring by identifying the relationship between one or more variables. The enquiry of this study is extracting patterns occurring in a sequence of adversary commands, rather than the regression problem of predicting the occurrence of commands without considering the sequence pattern. The clustering problem of grouping similar data points to form clusters does not align with the enquiry of this study. The clustering solution would cluster similar commands rather than identifying patterns between adversary commands occurring in a sequence.

The purpose of this study is to identify and extract patterns from a sequence of adversary commands. The machine learning problems were align with this study are classification and association. The classification problem encompasses training labelled data to form classes that can be utilised for decision and prediction.

The association problem addresses whether patterns of association exist between objects within a given dataset. This study investigates whether adversary commands can be classified based on their placement in a sequence of commands, allowing for the probability to be calculated for the observed commands occurring within a sequence.

This thesis attempts to identify whether there is an association of patterns between adversary commands in a sequence within a given dataset. However, labelled datasets are utilised within this study, deeming the association algorithms applied to the datasets as semi-supervised learning algorithms. As a training process is not conducted with the association algorithms and with unsupervised learning algorithms. Therefore, the association algorithms that have been applied in this study are categorised as semi-supervised.

To identify the classification and association machine learning algorithms that had been utilised to extract patterns within the sequence of adversary commands, the probability inferences that align with this study must be understood. The interpretation of the probability is essential since this study focuses on calculating the probability of a command occurring within a sequence based on the patterns extracted from the sequence of adversary commands. In the following section the interpretations of probability theories are elaborated.

2.3.1 Probability Theory

Probability theory provides the framework to address the uncertainties that arise within the domain of pattern recognition. Probability can be interpreted in four different ways based on the inference of the occurrence of an event through the epistemology of the interpretation (Howie, 2002). Epistemology is the nature of knowledge and how it is created. The four interpretations of probability that are presented in Table 2.3 are, classical, propensity, Bayesian and frequentist. The epistemology of each interpretation and an example of the four interpretations of probability are also presented in Table 2.3.

Table 2.3, Shows the four interpretations of probability. Along with the purpose and epistemology of the four interpretations of probability

Probabilistic interpretation	***Purpose***	***Epistemology***	***Example***
Classicalist (Mellor & Koslow, 2004a)	Prior knowledge is unknown, allowing for the probability of all outcomes to be equally possible.	The probability of an event occurring is evenly distributed between the possible outcomes.	The probability of choosing the correct multiple-choice answer from four possible answers (A, B, C, D) is 0.25.
Propensity (Mellor & Koslow, 2004b)	The physical property of an object or conditions can affect the probability of an event occurring.	The probability is based on the physical reality or property of an object in a single case occurrence.	A coin weighted on a single side would impact the probability when the coin is flipped.
Bayesian (Howie, 2002)	The context of the event occurring is known, allowing for a degree of confidence to be determined and quantified.	The Bayesian interpretation is the degree of confidence in an event occurring based on the subjective knowledge of the conditions or expectations.	Bayesian probability is commonly implemented as a spam filter, with particular known trigger words such as "*you have won"* flagging a possible spam email. The trigger words are determined based on prior knowledge on the subject.
Frequentist (Howie, 2002)	The frequency of an event occurring based on previous occurrences of that event.	The frequency probability interpretation is the probability of an event occurring based on past tests or trials, occurring. With an infinite amount of trials the results should converge and a true probability can be ascertained.	The frequency of the grocery shopper buying milk, eggs and bread within a single transaction can be utilised to predict grocery transaction patterns.

The classicalist interpretation of probability is not appropriate for this study. As the epistemology of the classicalist interpretation states, all outcomes have an equal probability of an event occurring (Mellor & Koslow, 2004a). This study recognises the probability of an adversary command occurring is established through calculating a sequence of adversary commands, rather than the probability of each command occurring in a single instance.

The propensity interpretation of probability is also not applicable to this study, as shown above in Table 2.3. The epistemology states, the physical property or reality of an object has an impact of the outcome of the event (Mellor & Koslow, 2004b). It is uncertain whether the physical property of the emulated SSH session of the adversary interacting with the honeypot could have impacted the outcomes of the sequence of commands utilised. Since the uncertainty is not tangible it cannot be confirmed as is required for the propensity interpretation of probability, it is not applicable for this study.

The Bayesian probability interpretation is the degree of confidence that can be determined and quantified for an event occurring based on knowledge of the conditions (Howie, 2002). The Bayesian probability interpretation is applicable to this study as the commands an adversary could utilise is based on the known commands that have occurred within that sequence. The Bayesian probability interpretation aligns with the classification problem solved by the machine learning algorithms (Howie, 2002).

The epistemology of the frequentist interpretation of probability is the occurrence of an event during a trial test, with an infinite amount of tests the results will converge to a true probability (Howie, 2002). The frequentist interpretation of probability is applicable to this study because the frequency of the sequence of adversary commands occurring could be used to calculate the probability of subsequent commands occurring. The frequentist interpretation of probability aligns with the association problem solved by the machine learning algorithms.

From examining the different probability interpretations, Bayesian and frequentist interpretations have been selected for this study. The Bayesian interpretation of probability aligns with the classification category of machine learning algorithms. While the frequentist interpretations of probability align with the association category of machine learning algorithms. The probabilistic classification algorithms that are based on the Bayesian theorem as well as the association algorithms that are based on the frequentist theorem are presented in the following sections.

2.3.2 Probabilistic Classification algorithms

There are various classification algorithms. As this study focuses on calculating the probability of sequences of commands occurring probabilistic classification algorithms had been selected. The probabilistic classification is a sub-group, within the classification category based on the Bayesian theorem. The probabilistic classification algorithms selected are the Naïve Bayes and the Markov chain algorithms.

The Bayesian theorem equation is presented in Eq 2.1 where P represents the probability of an instance occurring, the x denotes the prior probability or the hypothesis that is being investigated and y signifies the predictor or the conditional information based on a new piece of information.

$$P(x|y) = \frac{P(y|x)\,P(x)}{P(y)} \qquad \text{(Eq 2.1)}$$

2.3.2.1 Naïve Bayes algorithm

The Naïve Bayes algorithm is based on the Bayesian theorem of probability and is also a probabilistic classification machine learning algorithm (Howie, 2002). The Naïve Bayes algorithm requires labelled data used to train the classifier. The Naïve Bayes algorithm differs from the conventional Bayesian algorithms as each feature is treated as an independent feature. A class within the dataset is chosen as the predicted class used to train the Naïve Bayes classifier (Bishop, 2006). The training process encompasses the features that appear frequently in a class independently, this is known as the prior probability. The prior probability is then used to identify the likelihood of features appearing in a class. The training process is based on the predicted class. The trained classifier is then ready to be applied to a test dataset.

There are three types of Naïve Bayes algorithms, these are namely the Gaussian algorithm, the MultiNomial algorithm and the Bernoulli algorithm. The Gaussian Naïve Bayes algorithm is appropriate for normal data distribution and utilised the mean and standard deviation of a class to train the classifier. The MultiNomial Naïve Bayes algorithm is commonly utilised for filtering spam emails, as it is used for classifying multinomial distributed data such as text based data. While the Bernoulli Naïve Bayes algorithm is predominantly utilised for classifying binary value features (Bishop, 2006).

Within the domain of intrusion detection, the Naïve Bayes algorithm is commonly utilised as a base algorithm to, developing hybrid detection algorithms or alternatively testing the precision of the proposed hybrid detection algorithms (Aziz, Hanafi, & Hassanien, 2017; Goeschel, 2016; Kevric, Jukic, & Subasi, 2017; Xiao, Chen, & Chang, 2014). Through exploring literature there is evidence of the Naïve Bayes algorithm being used to filter SSH network flow data to identify adversarial activities, in particular, for SSH brute force authorisation attempts (Najafabadi, Khoshgoftaar, Calvert, & Kemp, 2015). A study conducted by Abd-Eldayem (2014), investigated an intrusion detection approach based on the Naïve Bayes algorithm, to detect adversarial activities that utilise the HTTP protocol. The experiments conducted have shown the Naïve Bayes algorithm is precise in classifying the adversarial activities based on network packet data, the experiments had been conducted using the aforementioned NSL-KDD dataset.

The Naïve Bayes algorithm has been utilised for intrusion detection purposes, predominantly for analysing network flow and network packet data. This study aims to contribute to the DPI domain utilising the Naïve Bayes algorithm. From examining existing literature, the Naïve Bayes algorithm has been utilised for analysing the network flow of SSH traffic rather than DPI. This study aimed to contribute to the domain by examining, whether the Naïve Bayes algorithm could be utilised as part of a DPI investigation to extract patterns within sequential SSH adversarial command interactions. There are limited studies that have been conducted on improving the efficiency of the Naïve Bayes algorithm

for classifying data. This study aimed to investigate whether an appropriately processed dataset could improve the efficiency of the Naïve Bayes algorithm.

2.3.2.2 Markov Chain algorithm

Since the data utilised for this study was labelled, the Markov chain supervised machine learning algorithm had been selected instead of the Hidden Markov Model (HMM), an unsupervised machine learning algorithm (Ali, Abbas, & Abbas, 2018). Markov chains are based on the Bayesian theorem of probability and are also a probabilistic classification machine learning algorithm. A Markov chain is a stochastic model that calculates the probability of the transition of states, from the current state to the next state occurring. A transition matrix is generated to predict the probability of a change in state, the columns represent the current states and the rows represent the next states. For this study, a state is the current command, a change in state is the probability of the next command in a sequence occurring. Training a Markov chain algorithm is adjusting the values in the transition matrix until they converge. The values in the columns equate to 1 as the probability is distributed between the possibilities in the next state occurring.

There is existing literature on the utilisation of the Markov chain algorithms for intrusion detection purposes. Markov chain algorithms have been applied to SSH network flow analysis (Hofstede, Hendriks, Sperotto, & Pras, 2014) and to detect stealthy brute-force SSH access attempts (Javed & Paxson, 2013). The Markov chain algorithms have also been studied to identify abnormal Internet applications utilising SSH sessions (Korczyński & Duda, 2014). The Markov chain algorithm implemented in a study conducted by Tapaswi *et al*. (2014) to predict the next IP address on a network an adversary will probe and move a honeypot to that particular IP address to gather further adversary interaction data. Tapaswi *et al*. (2014) show the Markov chain algorithm was successfully able to predict the next probeable IP address. Caselli *et al*. (2015), used Markov chains to predict the sequence of events within the Modbus traffic. The study successfully predicted anomalous traffic from a sequence of Modbus events. The scope of the study omitted the efficiency of the Markov chain to converge the transition matrix. Ali et al. (2018), conducted a review on the use of HMM for intrusion detection purposes and identified six prominent contributions to knowledge that can be undertaken within the domain. One of which is the need to efficiently train or converge the transition matrix for a HMM. The need to efficiently train or converge the transition matrix is also applicable to the Markov chain algorithm domain.

From examining literature, the Markov chain algorithm has been utilised within the intrusion detection domain for network flow analysis and packet inspection for services including SSH. Although applying the Markov chain algorithm to data extracted from a DPI investigation has been conducted on Modbus traffic, the commands predicted are Modbus service commands. This study examines whether the

Markov chain algorithm can efficiently extract patterns within sequential SSH adversarial commands interactions by means of DPI.

2.3.3 Association Rule Mining Algorithm

The association class machine learning algorithms are known as the association rule mining algorithms (Howie, 2002). This study investigates whether there is an association of patterns between adversary commands in a sequence within a given dataset. The association rule mining algorithms address whether patterns of association exist between objects within a given dataset and is based on the frequentist theorem of probability. The association rule mining algorithms examined in this section are the Apriori and Equivalence Class Transformation (Eclat) algorithms.

The frequentist theorem of probability is calculated using equation Eq. 2.2. To ascertain the true probability based on an infinite number of trials equation Eq. 2.3 should be utilised. *P* represents the probability of the events occurring based as seen in the given data and n_x represents the number of events occurring. n_t denotes the number of trials or tests conducted.

$$P(x) \approx \frac{n_x}{n_t} \tag{Eq. 2.2}$$

$$P(x) = \lim_{n_t \to \infty} \frac{n_x}{n_t} \tag{Eq. 2.3}$$

The Apriori and Eclat algorithms both calculate the probability of a sequence of objects occurring based on the frequency they appear in the given datasets. The extracted sequence of objects together with the probability of the occurrence is then formulated into rules. Each rule has three values of interest used to determine the probability of the extracted rule occurring; the values are support, confidence and lift. The support value determines the probability of a rule occurring in the given datasets. The confidence of a rule measures the probability of observing an object within a sequence, based on the presence of an initial object/s. The lift value given to a rule measures the dependency or independence of objects within an extracted rule.

2.3.3.1 Apriori algorithm

The Apriori algorithm is the prominent association rule mining algorithm and is based on the frequentist theorem of probability (Kotiyal *et al*., 2013). The frequentist theorem of probability is the occurrence of an event during a trial, with an infinite amount of tests the results will converge to a true probability.

In the context of this study, the events are the sequence of adversarial SSH commands. While the objects are the commands within the sequences. The Apriori algorithm is more suited for processing a greater volume of data compared to the Eclat algorithm (Kotiyal *et al*., 2013).

Research conducted by Khalili and Sami (2015), proposed an industrial intrusion detection approach to mitigate threats to cyber physical systems that utilised sequential patterns extracted by the Apriori algorithm to aid experts in identifying critical states. The study had shown the Apriori algorithm could be used to extract sequential patterns to monitor industrial processes. The study utilised network packet data which was utilised for inspection, while this study utilises DPI to identify patterns within adversarial sequence commands. A study conducted by Han-Wei, Huey-Min, and Wei-Cheng (2013), investigated the use of the Apriori algorithm to track adversaries transitioning through sequences of hosts to launch an attack. Data were extracted from network packets to determine the sequence of hosts. The Apriori algorithm had proven to be suitable for that study and provides evidence the algorithm would be applicable to this study.

Research has also been conducted in the utilisation of the Apriori algorithm outside of the cyber security domain. A study conducted by Jung and Chung (2015) applied the Apriori algorithm to the smart health service. The Apriori algorithm was applied to a set of images of a patient to generate bio sequential patterns of the patient. The bio-sequential patterns are then utilised to create a baseline and any deviation from the bio sequential patterns could suggest a possible emergency situation taking place. The study utilised surveillance technology to acquire the patient images. The study has given evidence of the use of the Apriori algorithm to establish bio-sequential patterns and could be applied to a sequence of adversary SSH commands to extract patterns.

A review of the current literature has shown the Apriori algorithm has been applied to the domain of cyber security as well as other knowledge domains. The review has shown the Apriori algorithm could extract sequential patterns from network flow and network packet data. This study aims to contribute to the domain of sequential pattern extraction from DPI data. In addition to extracting patterns from a sequence of adversarial commands, based on the association of commands within the sequence.

2.3.3.2 Eclat algorithm

Eclat algorithm is an association rule mining algorithm similar to the Apriori algorithm that is based on the frequentist theorem of probability (Kotiyal *et al*., 2013). The frequentist theorem of probability is the occurrence of an event during a trial test, with an infinite amount of tests the results will converge to a true probability. In the context of this study, the events are the sequence of adversarial SSH commands. The Eclat algorithm focuses on the frequency of objects occurring within a given dataset, by utilising a depth-first search technique. The algorithm is considered to be efficient at processing a

given dataset compared to the Apriori algorithm as well as processing data in real-time (Kotiyal *et al*., 2013).

Although there is a minimal application of the Eclat algorithm within the domain of cyber security, there is literature surrounding the application of the algorithm in other knowledge domains. A study conducted by Jin, Varadharajan, and Tupakula (2017) proposed a strategy based on the Eclat algorithm to detect malicious users that launch an attack that reports false location data on the Cognitive Radio Network. The proposed strategy based on the Eclat algorithm was proven to perform better than traditional approaches within the domain there providing evidence that the Eclat algorithm could be utilised for sequential adversarial commands. Additionally, a study conducted by Arincy and Sitanggang (2015), utilised association rule mining algorithms, including the Eclat algorithm. to extract a pattern between environmental conditions and forest fires. Suggesting the Eclat algorithm could be utilised for pattern extraction from sequential adversarial commands. A recent study conducted by Wang, Huang, Luo, Pei, and Xu (2018) utilised the Eclat algorithm to extract associations between hazards and the work environment in order to predict workplace hazards. The Eclat algorithm has also been utilised within the area of energy management. The study conducted by Jiang (2017) utilised the Eclat algorithm to identify associations between Global Horizontal Irradiance (GHI) and their associated variables in order to identify predictors that can be applied to a proposed model for photovoltaic installations. An additional study conducted by Jiang and Dong (2016), utilised the Eclat algorithm to identify associations between solar radiation and different meteorological variables to forecast global solar radiations.

The studies that have been examined involve the application of the Eclat algorithm within other knowledge domains outside of the cyber security domain. The studies have shown the Eclat algorithm could be utilised to extract associations between objects within a given dataset. Suggesting the Eclat algorithm could be utilised to extract patterns between sequential SSH adversarial commands. The application of the Eclat algorithm for pattern-based intrusion detection approaches for SSH is a contribution to the knowledge domain of cyber security and patter-based intrusion detection approaches.

The Naïve Bayes and Markov chain probabilistic classification algorithms and the Apriori and Eclat association rule mining algorithms have been chosen for this study. The chosen machine learning algorithms are applied to the sequential SSH adversarial commands datasets in order to extract patterns in this study. Since this study aimed to investigate a precise and efficient pattern-based intrusion detection approach enhancing the performance of the selected machine learning algorithms needed to be taken into consideration. In the following section, two approaches for enhancing the performance of machine learning algorithms are presented.

2.3.4 Enhancing Intrusion Detection Approaches

By examining literature there are two main approaches for enhancing machine learning algorithms for intrusion detection purposes. These are: 1. implementing hybrid algorithms and 2. feature selection. Hybrid algorithms which are developed by combining two or more algorithms. While feature selection is choosing relevant attributes within a dataset that will allow for additional information to be extracted by classifying the dataset based on choosing selected features.

2.3.4.1 Hybrid Algorithms

There have been studies conducted in enhancing intrusion detection approaches by developing hybrid algorithms and approaches to detect unauthorised activities. Kevric *et al*. (2017) developed a hybrid classification algorithm combining the random decision tree and the Naïve Bayes classifier. Experiments conducted on the NSL-KDD dataset showed the proposed hybrid algorithm outperformed the standard machine learning algorithms. Aslahi-Shahri *et al*. (2016) proposed a Support Vector Machine (SVM) and a Genetic Algorithm (GA) hybrid approach. The GA was used for feature selection while the SVM classified the data. The results obtained from applying the developed hybrid algorithm to the KDD Cup 99 dataset, showed the developed hybrid algorithm is more accurate at classifying network traffic compared to the standard machine learning algorithms tested. A study conducted by Soheily-Khah *et al.* (2018) developed a hybrid detection approach based on the K-means and Random Forest decision tree classifier. The K-means was utilised to pre-process the ISCX intrusion detection datasets followed by the results being processed by the Random Forest classifier. Results had shown the proposed algorithm was precise and efficient compared to other machine learning algorithms tested. In addition, studies have been conducted in applying fuzzy logic to association rule mining algorithms (Aburrous, Hossain, Dahal, & Thabtah, 2010; Changguo, Nianzhong, Tailei, Qin, & Xiaorong, 2009). Fuzzy logic allows more flexible segment boundaries, by giving the administrator control of defining the fuzzy set range associated with the boundary. The association rule mining algorithms that had fuzzy logic applied produced greater precision and efficiency compared to the standard association rule mining algorithms examined.

Hybrid algorithms have been proven to enhance the precision and efficiency of machine learning algorithms (Agrawal & Agrawal, 2015). However, the current study intended to utilise the machine learning algorithms to extract patterns from sequential SSH adversarial commands and compare the results between the reduced dataset to the respective full dataset. As such this study is not concerned with implementing enhanced versions of the chosen machine learning algorithms instead to enhance the performance of the existing algorithms.

2.3.4.2 Feature Selection

Studies have been conducted on improving the feature selection process to enhance intrusion detection approaches. Feature selection is choosing relevant attributes within a dataset that will allow for additional information to be extracted by classifying the datasets based on the selected features. Studies within the relevant literature have focused on improving feature selection as part of developing hybrid approaches to enhance intrusion detection approaches. De la Hoz, De La Hoz, Ortiz, Ortega, and Prieto (2015) suggested using the probability based Self-Organising Maps (SOMs) and using Principle Component Analysis (PCA) and Fisher Discriminant Ratio (FDR) for selecting features. The experiments conducted suggested the proposed feature selection approach applied on the NSL-KDD dataset was precise compared to the results obtained from standard algorithms. Gauthama Raman, Kirthivasan, and Shankar Sriram (2017) focused on Rough Set Theory (RST) to extract additional data from a dataset and proposed Rough Set Hyper-graph (RSHGT) as a solution for extracting an optimised feature subset. Experiments had been conducted on the KDD Cup 99 dataset, the proposed RSHGT was evaluated by applying the selected features on chosen classifiers then comparing the results to other feature extraction techniques using the same chosen classifiers.

Aminanto, Choi, Tanuwidjaja, Yoo, and Kim (2018), identified the optimal feature to detect an impersonator on a Wi-Fi network using machine learning algorithms. The results show by identifying the optimal features the performance of that machine learning algorithms can be enhanced. This suggests selecting the optimal feature for the current study can enhance the performance of the machine learning algorithms. The feature selection process can take place in the pre-processing phase of this study. The study conducted by Soheily-Khah *et al*. (2018) presented the pre-processing procedure that had been implemented on the ISCX dataset. The processing procedure implemented was developed to convert raw traffic and separate the data into different traffic types (normal and abnormal). That particular study converted the data from nominal to numeric.

The pre-processing phase is a critical process when applying machine learning (Malley *et al.*, 2016). In the pre-processing phase, the datasets are massaged and prepared to be applied to the selected machine learning algorithms. However, there is a combination of steps that can be taken in the pre-processing phase depending on the requirements of this study and the data collected. Within literature, there are variations of the pre-processing phase (Bramer, 2013; García, 2015; Hackeling, 2014). Hence, after considering the combination of possible steps within the pre-processing phase and the requirements of this study, a pre-processing phase was developed for this study (presented in Section 4.2). The pre-processing procedure developed processes, the acquired SSH honeypot datasets to exhibit the sequence of adversary commands for a unique session.

The classification and association problems that are solved by the machine learning algorithms, align with the current study. In order to identify the classification and association machine learning algorithms that should be utilised, the four main probability inferences were examined. The Bayesian and frequentist interpretations of probability were selected for this study. The Naïve Bayes and Markov chain probabilistic classification algorithms, that are based on the Bayesian theorem of probability had been chosen. Along with the Apriori and Eclat association rule mining algorithms, that are based on the frequentist theorem of probability, had also been chosen for this study.

2.4 Conclusion

Traditionally intrusion detection approaches are either signature or misuse based or anomaly based detection approaches. An intrusion detection approach should precisely and efficiently detect adversaries on the network in a timely manner to avoid assets being compromised. The current study investigated whether patterns extracted from adversarial SSH commands can be utilised as a pattern-based intrusion detection approach. The data for this study has been acquired from three medium interaction research honeypots that emulate the SSH protocol. Since the open-source intrusion detection datasets had been deemed unsuitable for this study as the adversary command interaction data is absent.

As the focus of this study was the use of adversary command patterns for intrusion detection purposes, DPI, was required to ascertain the adversary command data. The examination of existing studies show there has been research conducted in the domain of adversary based intrusion detection approaches utilising DPI. Although there are gaps in the knowledge domain this thesis aims to contribute in terms of pattern identification within a sequence of adversary interaction commands. By investigating whether more precise patterns can be extracted efficiently from a reduced dataset by selected machine learning algorithms compared to the patterns extracted from the respective full dataset.

Machine learning is the application of learning algorithms to extract knowledge from data to determine patterns between data points and in turn make predictions. The Naïve Bayes and Markov chain probabilistic classification algorithms, that are based on the Bayesian theorem of probability, were chosen along with the Apriori and Eclat association rule mining algorithms, that are based on the frequentist theorem of probability. For the current study a pre-processing procedure that encompasses feature selection and data reduction are deployed, to be applied to the acquired datasets prior to the application of the chosen machine learning algorithms.

The Naïve Bayes and Apriori algorithms have been utilised for intrusion detection purposes, predominantly for analysing network flow and network packet data. The current study aimed to contribute to the domain by examining whether the Naïve Bayes and Apriori algorithms could be

utilised for DPI to extract patterns within sequential SSH adversarial command interactions. The Markov chain algorithm has been utilised to extract data by means of DPI for Modbus traffic. However, the Markov chain algorithm has not been applied to extract patterns within sequential SSH adversarial commands interactions by means of DPI. This study also researched whether reducing a dataset can improve the efficiency of training or converging the Markov chain transition matrix, further contributing to the knowledge domain. The Eclat algorithm has predominantly been utilised in studies outside of the cyber security domain. Those studies had shown the Eclat algorithm can be utilised to extract associations between objects within a given dataset. Suggesting the Eclat algorithm could be utilised to extract patterns between sequential SSH adversarial commands. The application of the Eclat algorithm for pattern-based intrusion detection approaches for SSH would contribute to the Eclat application in the cyber security domain. The research methodology and research design are discussed in the following Chapter 3.

3 Research Methodology and Design

3.1 Research Methodology

In order to define the research methodology for this study the philosophical assumption also known as the research paradigm, has to be established. Williamson and Johanson (2013) state a research methodology is a framework consisting of a set of principles that are underpinned by the chosen research paradigm. The research paradigms that are discussed are; positivism, post-positivism, interpretivist and critical theory (Creswell, 2014; Williamson & Johanson, 2013).

The aim of this study is established through the ontology and epistemology of this study. Ontology is concerned with this study of reality and the existence of things. Posing the questions, when is the thing real and how do these things interact? Epistemology is nested in the ontology of the research. It is concerned with the nature of knowledge and how it is created. Posing the question, what is the relationship between the research and the researcher? Table 3.1, shows the ontology and epistemology for each of the four research paradigms along with the reason for conducting the research and modes of inquiry (Creswell, 2014; Mackenzie & Knipe, 2006; Williamson & Johanson, 2013)

The positivist research paradigm is also referred to as rationalism or realism. Both positivist and post-positivist use deductive empirical techniques to verify hypotheses through testing. Discoveries are made using quantitative methods of inquiry and validation. The common research designs associated with positivism and post-positivism are experimental design or survey techniques. The difference between positivism and post-positivism is the differences in the core principle of each paradigm. The principle of positivism is to identify the absolute truth through measurements and observations. Hypotheses need to be proven as the absolute truth, else the hypotheses are rejected. Whereas, the principle of post-positivism is the absolute truth is unattainable, as not all variables can be controlled or accounted for. For this reason, a wider critical analysis should be conducted. Hypotheses are proven through the correctness of the statement through scientific inquiry rather than obtaining absolute truth.

The interpretative research paradigm involves investigating the meaning and experiences of human interactions. The interpretative research paradigm is also linked with constructivism and phenomenology, major differences are in the purpose of the research conducted. Interpretivism focuses on how the researcher or participants perceive the surrounding world. Constructivism focuses on the interaction of the participants and the surrounding world. Phenomenology is focused on the experiences of the participants and the surrounding world, and qualitative means of inquiry are commonly used with research techniques including; surveys, interviews and observing the participants. Theories are predominantly developed after the data analysis phase is complete. Regarding the current study, theories

are tested through the developments of hypotheses and tested using experiments before the data analysis phase. Thus, the research paradigm for this study does not align with the interpretative research paradigm.

Table 3.1, Shows the reason for research, ontology, epistemology, mode of inquiry for the four research paradigms. Adapted from Creswell (2014), Mackenzie and Knipe (2006) and Williamson and Johnson (2013)

Paradigms	***Positivist***	***Post-positivist***	***Interpretive***	***Critical Theory***
Reason for research	To discover the absolute truth to predict and describe events.	To discover the correctness of the truth to predict and describe events.	To understand and describe human interactions with the surrounding world.	To empower society to challenge and change social divides.
Ontology	There is a reality 'out there' waiting to be discovered, regardless of the researcher	There is a reality 'out there' but it's beyond the control of the researcher	Reality is conditional on the interpretations of experiences by people.	Reality is socially constructed and perceived objectively.
Epistemology	Knowledge is unbiased, it is real when empirically true and repeatable.	Reality can only be approximated. Knowledge can be obtained through correctness using empirical learning. Bias can impact knowledge if not controlled and maintained.	Knowledge is subjective based on the participants of the research and the interactions they have with the surrounding world.	Knowledge is subjective based on power and justice related to social divides. Knowledge can be used to empower the social constraints.
Mode of inquiry	• Laboratory experiments • Quasi-experimental • Sample survey	• Sample survey • Quasi-experimental	• Ethnographic case study • Survey • Interview • Observation	• Critical ethnography • Collaborative inquiry • Survey • Focus groups • Interview • Observation

The critical theory research paradigm is focused on social empowerment. Although similar to interpretivism the main difference is the defined scope of the research. Interpretivism, is seen to have a broad objective, whereas the critical theory is concerned with an in-depth exploration of particular issues arising from social divides. Investigations are conducted using a qualitative means of inquiry. The focus of critical theory research is on social change through the understanding of the research problem. Therefore, the critical theory research paradigm does not align with the current study as this study focus is on discovering knowledge through identifying patterns within a sequence of commands utilised by adversaries.

Post-positivism was selected as the research paradigm for this study as the core principles of scientific inquiry to prove the correctness of the hypotheses aligns with this study. The ontology of this study

recognises the absolute truth could exist but may not be attainable by the researcher. Knowledge is discovered through measurements and observations using empirical techniques as the epistemology. Knowledge is discovered through identifying patterns within a sequence of commands utilised by adversaries, the absolute truth is unattainable. However, the correctness of the hypotheses can be tested, hence a post-positivist research paradigm was selected. Therefore, this PhD study has taken a quantitative method with an experimental research design.

Experimental research design uses a quantitative means of inquiry. Experiments can be conducted in either a laboratory setting or as field-based studies (Jackson, 2012). Laboratory experiments are based in precise control environments. The researcher has greater control over variables and influencing factors compared to field experiments. Field experiments are conducted in a 'real' setting subsequently, the researcher has less control over variables and influencing factors. This study used a field based experimental design due to the lack of control of some variables and factors. The current study focused on data collected from real adversarial interactions. Thus, the datasets had been acquired from public facing honeypots capturing legitimate adversarial activities as opposed to deploying internal facing honeypots to capture simulated activities. Therefore, there was a lack of control over the volume and adversary interactions and the resultant data recorded by the honeypots. The lack of control over some variables and factors fit within the quasi-experimental research design frame. The quasi-experimental design acknowledges there can be a lack of control over extraneous variables and factors of the research.

Quasi-experimental research designs can either be; non-equivalent comparison group design, interrupted time-series design or regression discontinuously design (Christensen, Johnson, & Turner, 2011; Jackson, 2012). Non-equivalent comparison group design encompasses both an experimental group and control group, the data assigned to groups are randomly selected. However, they are not identical instead the groups are comparable. With the interrupted time-series design measurements are taken before, during and after testing. Measurements are taken several times and can be taken over an extended period of time. Whereas, the regression discontinuous design allocates groups based on assignment measurements prior to testing. The focus is on any irregularity in the groups plotted regression line after testing.

After evaluating the three quasi-experimental designs a non-equivalent control group pretest-posttest design was selected. As the experiments in this study evaluate and analyse the patterns extracted from sequential adversarial patterns in a full dataset to a reduced dataset after appropriately pre-processing (data treatment) the dataset. In a pretest-posttest, the whole dataset is tested before and after applying the data treatment framework and the results are compared. A posttest was not chosen as the results are only analysed after the treatment is applied, leaving no prior test comparison results (Christensen *et al*., 2011).

The underpinning research paradigm for this study is a post-positivist quantitative approach. This is in conjunction with a field experimental research design using a quasi-experimentation in a non-equivalent control group pretest-posttest design.

3.2 Research Questions

After reflecting upon the literature surrounding the area of study, three research questions were formulated. The hypotheses associated with each research question have been developed to test, evaluate and answer the research questions posed. The objective of this study is to investigate whether selected machine learning algorithms can yield more precise patterns efficiently from the reduced sequence of commands datasets compared to their respective full datasets. For this study precision is in reference to correctly identifying patterns in sequential adversarial commands within the given datasets. Furthermore for this study efficiency is in reference to the processing time for the selected machine learning algorithms to extract the patterns from the given datasets.

RQ1: Is there an increase in the number of patterns extracted from the reduced datasets by the machine learning algorithms, compared to the number of patterns extracted from their respective full datasets?

H1: More class patterns can be extracted from the reduced sequence of commands datasets, compared to those extracted from their respective full datasets by the selected probabilistic classification algorithms.

H2: More patterns can be extracted from the rule sets of the reduced sequence of commands datasets, compared to those extracted from their respective full datasets by the selected association rule mining algorithms.

RQ2: Are the extracted patterns from the reduced datasets by the machine learning algorithms overall more precise, compared to those extracted from their relevant full datasets?

H3: The class patterns extracted by the probabilistic classification algorithms from the reduced sequence of commands datasets are more precise, compared to those extracted from their respective full datasets.

H4: The patterns extracted from the rule sets by the association rule mining algorithms from the reduced sequence of commands datasets are more precise, compared to those rule sets extracted from their respective full datasets.

RQ3: Are the machine learning algorithms more efficient at extracting patterns from the reduced datasets, compared to the processing time of their respective full datasets?

H5: The probabilistic classification algorithms are more efficient at extracting patterns from the reduced sequence of commands datasets, compared to their respective full datasets.

H6: The association rule mining algorithms are more efficient at extracting patterns from the rule set for the reduced sequence of commands datasets, compared to their respective full datasets.

3.3 Research Variables

When conducting experiments to test the hypotheses, the research variables associated with this study must be identified as either; dependent, independent, controlled, and confound variables. Within this section, the research variables for this study are presented.

3.3.1 Dependent Variables

The dependent variables that are measured and observed when testing hypotheses are stated below.

- DV1: Precision as stated in Section 3.5, of each machine learning algorithm identifying patterns in the sequence of commands utilised by adversaries.
- DV2: Time in milliseconds (efficiency), for each algorithm to calculate the probability of a chain of adversary commands occurring. The tictoc *R* package was chosen to measure the time in milliseconds as stated in Section 3.5.
- DV3: The patterns within the classification of the datasets extracted by the probabilistic classification algorithms.
- DV4: The patterns within the rule sets extracted by the association rule algorithms from the datasets.
- DV5: The length and commands within a chain utilised by an adversary during a session.

3.3.2 Independent Variables

Independent variables cause change when interacting with other variables in this study. In this study, the experimental datasets are the independent variables. Since the changes in the patterns extracted from the full and the corresponding reduced sequence of commands datasets by the machine learning algorithms.

- IV1: Experimental datasets:
    - The full sequence of commands datasets: The full length of the sequence without clustering the commands.

- The reduced sequence of commands datasets: The reduced sequence of commands. By clustering commands together based on the function performed by the given command.

3.3.3 Controlled Variables

To maintain the validity of this study when testing the hypotheses, control variables must be isolated and kept constant, to avoid this study being compromised affecting the dependent and independent variables. To facilitate outcome this the following are required:

- ContV1: Computer setup, all experiments and tests should be conducted on the same computer device (elaborated in Section 3.6.1). The computer device was also isolated from other networks ensuring the device was kept at a constant throughout the experimental phase.
- ContV2: Software used, the software applications and associated packages used to conduct the experiments throughout this study (elaborated in Section 3.6.2) had been kept constant. No updates, application settings or application configurations were altered once the experimental phase was initiated.
- ContV3: The machine learning algorithms:
    - Probabilistic classification algorithms: Naïve Bayes and Markov chains algorithms.
    - Association rule mining algorithms: Apriori and Equivalence Class Transformation (Eclat) algorithms .
- ContV4: Implementation of the machine learning algorithms: The same implementation and the parameters set for each of the four machine learning algorithms have been maintained throughout the experimental phase. This maintenance ensured the validity of the results when testing the hypotheses.

3.3.4 Confounding Variables

Confound variables are extraneous variables that can impact the dependent and independent variables. Below is the confounding variable that was outside the control of the researcher:

- ConfV1: The acquired honeypot datasets: Since this study was focused on investigating 'real' adversary interactions, the datasets had been acquired from external sources that deployed publicly facing honeypots. The length of the sequence and the commands utilised by an adversary within a session was outside the control of the researcher.

3.4 Research Procedure

The research procedure developed for this study is based on the Cross Industry Standard Process for Data Mining (CRISP-DM) framework (SPSS, 2000). The CRISP-DM framework is a common framework utilised for machine learning projects. The CRISP-DM framework consists of six phases;

business understanding phase, data understanding phase, pre-processing phase, modelling phase, evaluation phase and deployment phase.

1. *Business understanding:* The business understanding phase is the initial phase of the CRISP-DM framework. Within this phase, the objectives of the project are identified and understood. The objectives identified are referred to throughout the other phases to ensure the project requirements are met.
2. *Data understanding:* In this phase, an initial exploration of the dataset or datasets is conducted to determine if the data is suitable for the project.
3. *Data pre-processing:* This phase encompasses multiple steps to clean, format and reduce the data preparing it for the modelling phase. Important features that are associated with the objectives of the project are selected, this is known as feature selection. The aim of the data preparation phase is to structure the data to processed to the modelling phase.
4. *Modelling:* The modelling phase is when the prepared data is applied to machine learning algorithms to extract knowledge.
5. *Evaluation:* In this phase, an analysis of the outcomes is conducted and the precision of the model's outcomes are evaluated and verified, to confirm the objective of the project has been achieved.
6. *Deployment:* The evaluated model is prepared for production concluding the CRISP-DM process.

The research procedure depicted in Figure 3.1 has been designed specifically for this study and consists of five phases, unlike the CRISP-DM framework. The development phase of the CRISP-DM framework was not applicable for this study as the model developed for the study is not for production purposes. The five phases designed specifically for this study are; project understanding phase, data understanding phase, pre-processing phase, experimental phase and finally the evaluation and analysis phase. The preliminary analysis is comprised of phase one, phase two and phase three. The project understanding phase is similar to the business understanding phase in the CRISP-DM framework however the phase has been adapted to reflect a research project as opposed to a business project since the goal of the current study is to add knowledge to the research domain. The data understanding phase from the CRISP-DM and Figure 3.1 share the same goal of conducting initial exploration and analysis of the acquired honeypot datasets. In order to determine the suitability of the data to be utilised in this study. The pre-processing phase is a critical phase in the research procedure because the datasets are treated resulting in the full and the corresponding reduced sequence of commands datasets generated for each of the three acquired honeypot datasets. The experimental phase in Figure 3.1, is the phase the machine learning algorithms are applied to extract patterns from the sequence of commands datasets. In the fifth phase, the patterns extracted from the sequences of commands datasets are evaluated and analysed.

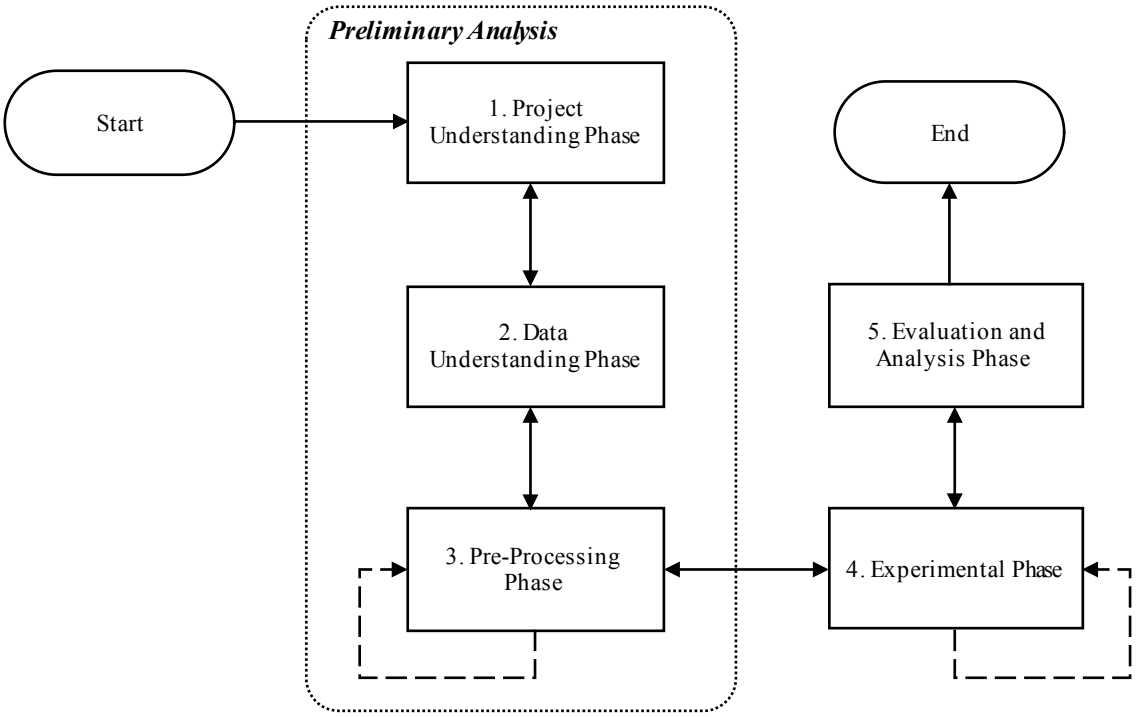


Figure 3.0.1, Illustrates the research procedure designed for this study consisting of five phases.

The process proceeds through all five phases. However, if the desired outcome of the phase is not achieved the process iterates back to the previous phase. The desired outcome for each phase must be achieved before proceeding further as each phase is designed to assist in answering the research questions posed in Section 3.2. Phase three and phase four are critical phases of the research procedure if all the steps in each phase are not completed the process is repeated because within these phases the pre-processing procedure is conducted and the experiments take place. The phases in the research procedure depicted in Figure 3.2 are elaborated further in the following sections.

3.4.1 Phase One, Project Understanding Phase

Phase one is the initial phase of the research procedure, in this phase the objectives of this study are defined as illustrated in Figure 3.3. The goal of this study is to be determined by identifying the gap in the knowledge this study intends to fill and is presented in Chapter 2. The resources available and the scope of this study are identified to ensure the goal of this study is attainable. The research questions that underpin the research can be answered through experimentation. The associated hypotheses should be determined as they are used to evaluate the experiment performed by ensuring the research questions can be answered. The research questions and associated hypotheses are presented in Section 3.2 and subsequently, the research variables are provided in Section 3.3.

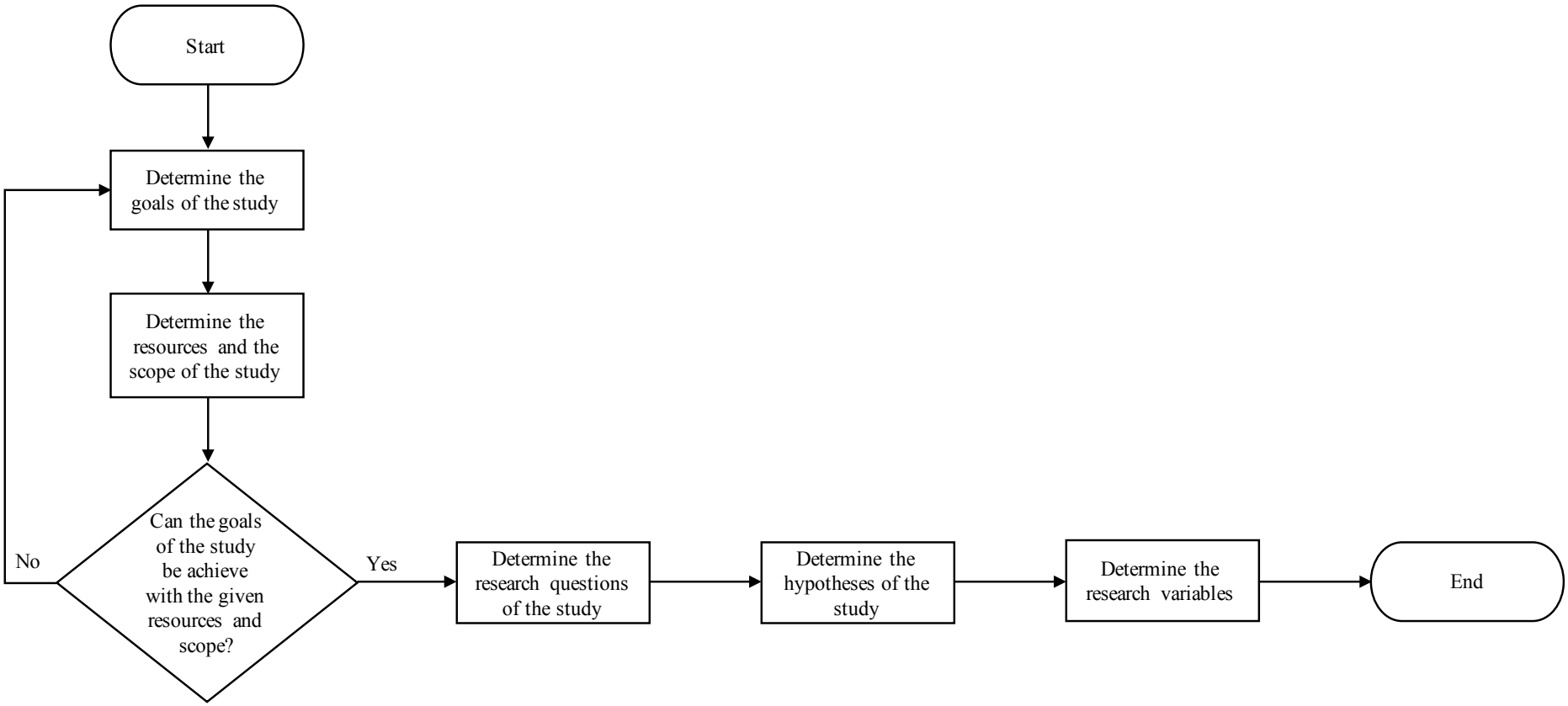


Figure 3.0.2, Illustrates a flowchart showing the process of phase 1, the project understanding phase

3.4.2 Phase Two, Data Understanding Phase

The data understanding phase is where the three honeypot datasets acquired for this study are explored and analysed to determine if they are suitable for this study as shown in Figure 3.4. If the datasets are not in a readable format they need to be reformatted to allow for initial exploration to be conducted. The initial exploration of the datasets consists of identifying the tables or files within the dataset and the relationship between each table or feature of the collected data. Upon completion of this step, the analysis of the dataset is to be conducted by identifying if the required features are seen in the dataset, deeming them suitable for this study. However, if the required features are not seen another dataset would need to be acquired. The application of the data understanding phase is presented in Section 4.1.

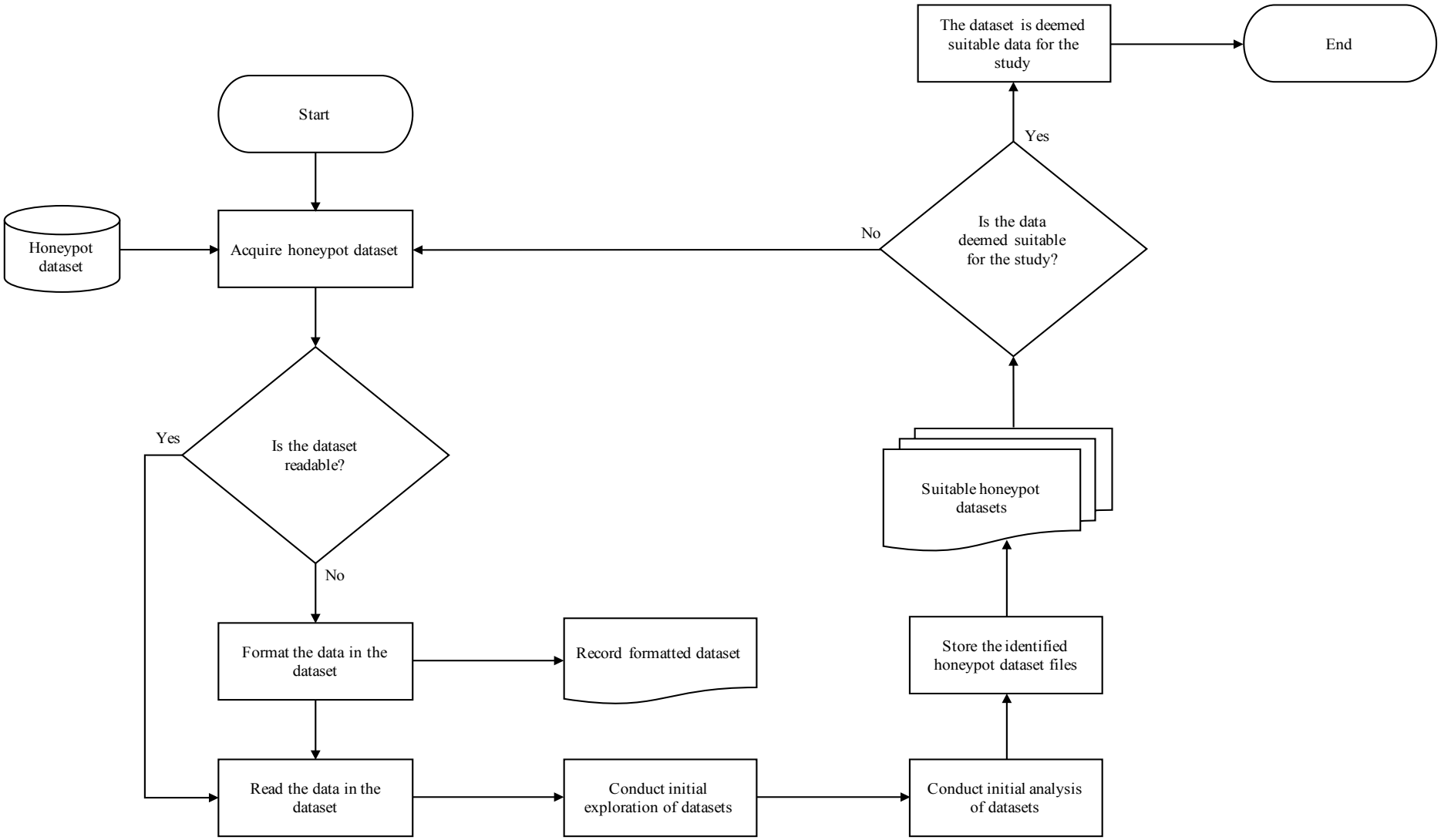


Figure 3.0.3, Illustrates a flowchart showing the process of phase 2, the data understanding phase.

3.4.3 Phase Three, Pre-Processing Phase

The pre-processing phase is a critical phase in the research procedure because in this phase the reduced datasets are generated and all the datasets are prepared to be applied it the selected machine learning algorithms (Malley *et al.*, 2016). The three dataset files from the previous data understanding phase are massaged until the full, and the corresponding reduced sequence of commands datasets, are produced as illustrated in Figure 3.5. This phase is initiated by selecting an acquired honeypot dataset. The data is filtered by removing redundant, noisy and incomplete data. Once the dataset has been filtered the process then proceeds to the data integration step. The tables or files are merged and attributes are selected to provide a holistic view of the data as well as resolving conflicting data.

This is followed by the data transformation step where the data is formatted by transforming the source data, to allow selected machine learning algorithms to process the sequence of commands datasets. Additionally, within the data transformation step, the unique commands are extracted and assigned a unique key value. The process does not continue until the data transformation step has been executed corrected.

Upon the completion of the data transformations step, a duplicate for the full sequence of command dataset is generated, with the data reduction step only applied to the duplicated dataset. The purpose of the data reduction step is to represent the full dataset using a reduced sequence of commands dataset. The reduced dataset is produced by clustering commands together based on the function of a given command. Thereafter each command cluster is assigned a unique key value. The process iterates back before an acceptable reduced dataset is recorded. The integrity of the full and associated reduced datasets are ensured by generating and storing the MD5 and SHA256 hash sum.

The final step in the pre-processing phase is the data wrangling step. In the data wrangling step, the data type is converted according to the machine learning algorithm selected. Therefore the step is conducted in the experimental phase before a chosen machine learning algorithm is applied to a dataset. Once all three honeypot datasets have been pre-processed resulting in three full datasets and their associated reduced datasets, the process proceeds to the next phase. Section 4.2 illustrates the pre-processing phase for the three acquired honeypot datasets.

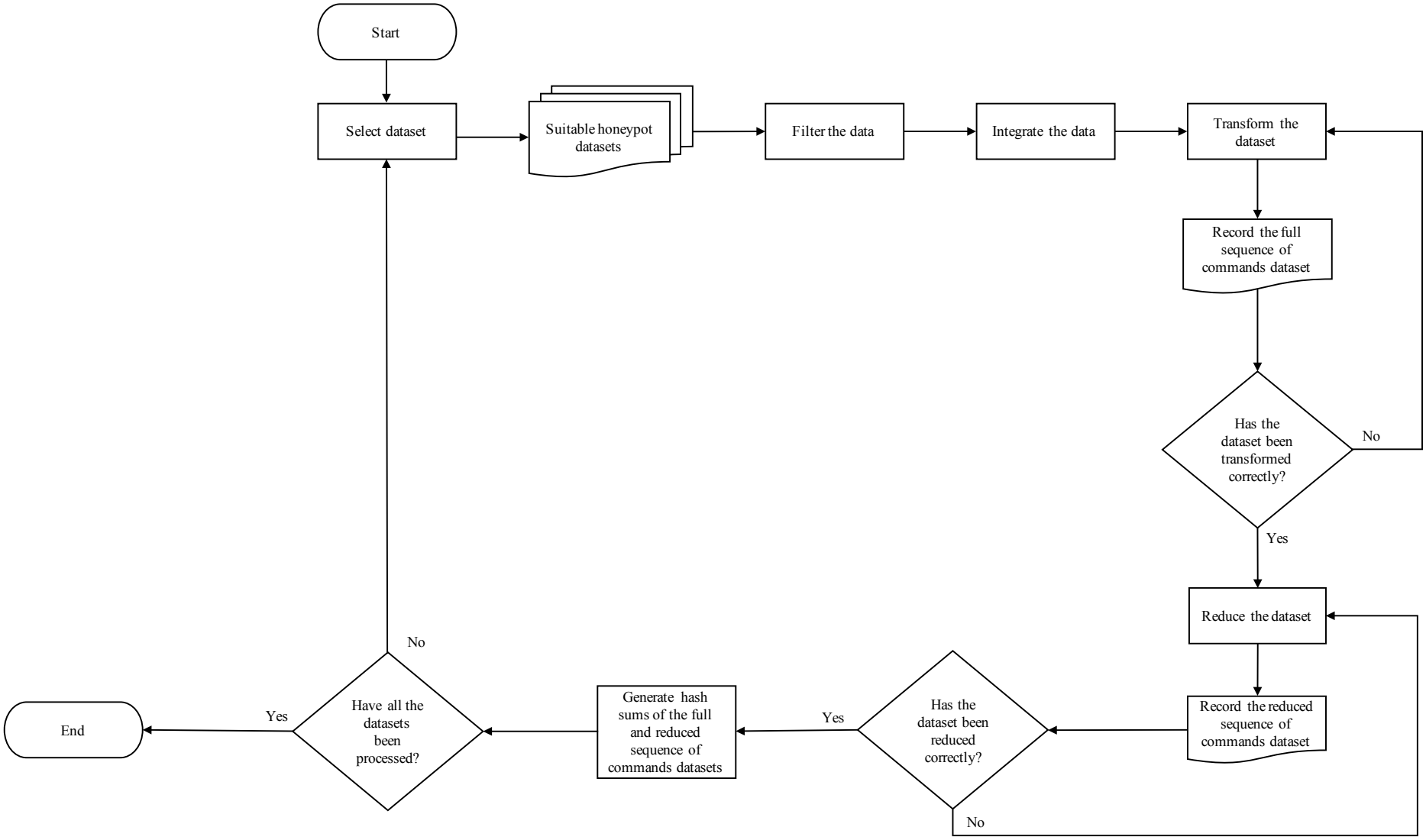


Figure 3.0.4, Illustrates a flowchart showing the process of phase 3, the pre-processing understanding phase.

3.4.4 Phase Four, Experimental Phase

In the experimental phase illustrated in Figure 3.6, the four machine learning algorithms are applied sequentially to the three full and their associated reduced sequence of commands datasets from the three-honeypot datasets. The process starts by selecting a pre-processed dataset and a machine learning algorithms for pattern extraction. The four machine learning algorithms are; Naïve Bayes, Markov chain, Apriori and Equivalence Class Transformation (Eclat). The data wrangling step is the final step of the pre-processing phase, the data type of the dataset is verified to suit the selected machine learning algorithm.

Once the appropriate data type has been set, if the Naïve Bayes classification algorithm had been selected, the chosen dataset is split randomly using *R* into a 70% train and 30% test set. However, if an association rule algorithm or the Markov chain algorithm was selected, the chosen algorithm was applied to the whole dataset without splitting it into a train and test set. Association rule algorithms extract rules based on the complete dataset and evaluation is based on the strength of each rule. Therefore, the need for a train and test set method of evaluation is not necessary instead the extracted rules are analysed based on the precision of each rule extracted shown in selection 3.5. The Markov chain algorithm was also applied to the complete dataset as the associated transition matrix is used to predict the probability of the change in state. For this study, a state is the current command, a change in state is the probability of the next command in a sequence occurring. Therefore, the need for train and test set method of evaluation thus, the evaluation was conducted on the transition matrix generated presented in Section 3.5.

Once the selected algorithm has been applied to the dataset, the test results are recorded. The process applied to the chosen dataset process had been repeated 100 times for each of the four machine learning algorithms to ensure precision measurements are repeatable, before proceeding to select another dataset. Once all datasets have been applied to the four machine learning algorithms, the outputs for all the experiments are collated, ready for the evaluation and analysis phase.

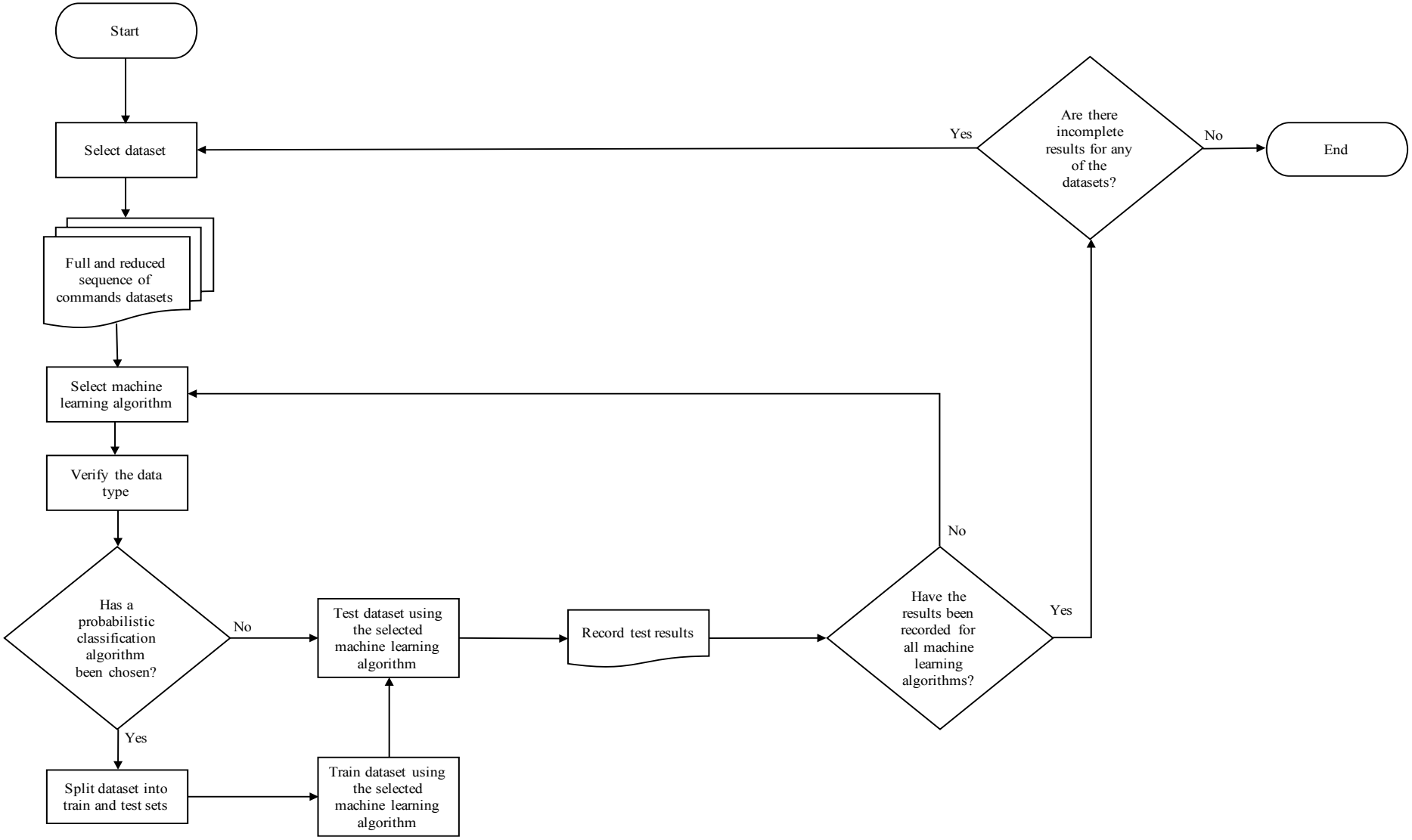


Figure 3.0.5, Illustrates a flowchart showing the process of phase 4, the experimental phase.

3.4.5 Phase Five, Evaluation and Analysis Phase

In this phase, the outcomes from the tests conducted in the experimental phase are evaluated and analysed as shown in Figure 3.7. The first step is to select the test results of the full and the corresponding reduced sequence of commands datasets.

Next, the extracted patterns from the datasets by the machine learning algorithms are compared. The comparisons are made between the patterns extracted by the probabilistic classification algorithms (referring to *H1*). Followed by comparing patterns in the extracted rule sets by the association rule mining algorithms (referring to *H2*).

Thereafter, the precision of the extracted patterns is evaluated. The precision measurements for Naïve Bayes algorithm experiments are determined using the confusion matrix (referring to *H3*). To evaluate the precision (positive predictive rate), accuracy rate, sensitivity (recall or true positive rate) and F1 score and error rate are calculated as outlined in Section 3.5. The precision measurements for the Markov chain experiments are based on the transition matrix generated. The precision of the generated transition matrix is established using the mean and standard deviation of the standard error matrix, lower endpoint matrix and upper endpoint matrix.

The patterns within the extracted rules by the association rule mining algorithms are evaluated for the precision (referring to *H4*). The precision of the extracted rules are calculated using the; confidence, support and lift as outlined in Section 3.5. Once the precision of the extracted patterns from the full and the corresponding reduced datasets had been calculated and evaluated the efficiency of the machine learning algorithms are calculated.

The efficiency of the machine learning algorithms to process the datasets are measured in milliseconds as stated in Section 3.5 (referring to *H5* and *H6*). Once all the extracted patterns of the full, and their associated reduced datasets, had been evaluated the outcomes are analysed. The final step is to report the findings and observations from the analysed outcomes.

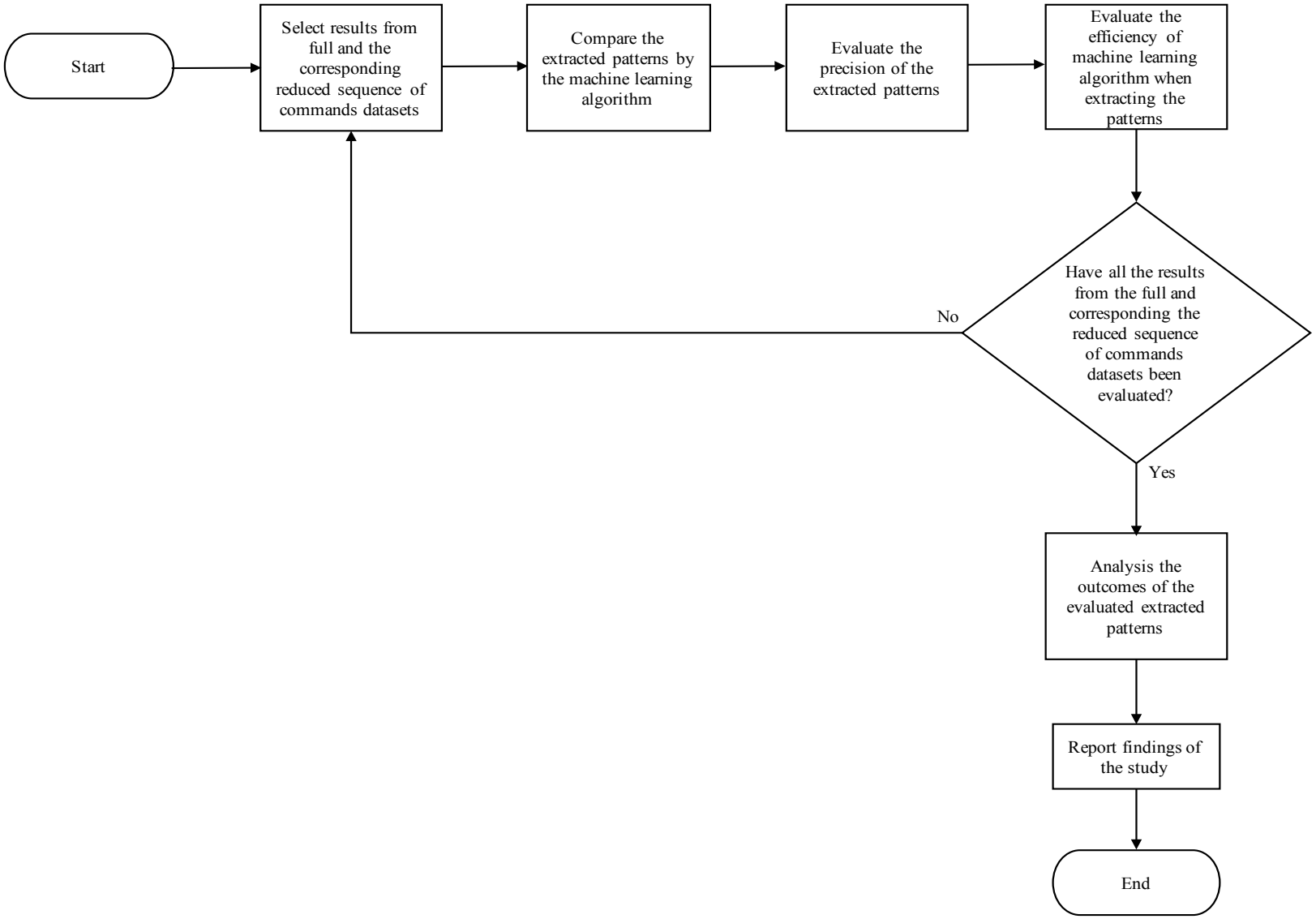


Figure 3.0.6, Illustrates a flowchart *showing the process of phase 5, evaluation and analysis phase*

3.5 Data Analysis

Various measurements can be used to evaluate the precision of machine learning algorithms. The Naïve Bayes algorithms commonly use the confusion matrix to measure the precision of the classification produced by the algorithms. A confusion matrix is a table that shows the instances of correct and incorrect objects predicted by the algorithm as illustrated in Table 3.2. The confusion matrix was used to calculate the; precision (positive predictive rate, PPR), accuracy rate, sensitivity (recall or true positive rate), F1 score and error rate.

Table 3.2, Shows a confusion matrix used to measure the performance of a classification learning algorithm.

		Predictive	
		Positive	*Negative*
Actual	*Positive*	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Precision is used to measure the correctness of the classification algorithm when classifying data. The accuracy was used to calculate the probability of the classified data truly being classified correctly by the algorithm. Sensitivity is also known as the recall, and the true positive rate is the number of objects correctly classified as positive the F1 score measures the predictive probability of the classification produced by calculating a balanced mean between the precision and sensitivity. The error rate measures the rate of incorrectly classified data.

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

(Eq. 3.1)

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} = 1 - Error\ Rate$$

(Eq. 3.2)

$$Sensitivity\ (recall) = \frac{TP}{(TP + FN)}$$

(Eq. 3.4)

$$F1\ score = \frac{2 \cdot precision \cdot sensitivity}{precision + sensitivity}$$

(Eq. 3.5)

$$Error\ Rate = \frac{FP + FN}{TP + TN + FN + FP} = 1 - Accurancy$$

(Eq. 3.5)

The precision of the Markov chain algorithm applied to the full and their associated reduced datasets are measured based on the transition matrix generated. The mean and standard deviation of the standard error matrix, lower endpoint matrix (lower limit) and upper endpoint matrix (upper limit) are used to evaluate the generated transition matrix by the Markov chain. A transition matrix is used to predict the probability of the change in state. For this study, a state is the current command, a change in state is the probability of the next command in a sequence occurring. The standard error matrix shows the error in the state change of the transition matrix. The lower endpoint matrix is the lowest bound of the confidence interval set at 0.95 for the values with the transition matrix. While the upper endpoint matrix is the upper bound for the confidence interval set at 0.95 for the values with the transition matrix. The mean and standard deviation values for the lower endpoint and the upper endpoint are calculated and compared. The mean is the average value of a set of variables and the standard deviation signifies the spread or divergence of the variables from the mean.

$$Mean = \overline{\chi} = \frac{(\sum \chi \mathrm{i})}{\eta}$$

(Eq. 3.6)

$$\text{Standard deviation} = \sigma_{\overline{\chi}} = \frac{\sigma}{\sqrt{\eta}}$$

(Eq. 3.7)

The precision of rules extracted by association rule mining algorithms are measured by the confidence, support and lift of each rule extracted. Confidence is determined by how frequent an object will appear in a command execution sequence that contains a subsequent object. The support is how often the rule appears in each dataset and the lift indicates if objects are independent or dependent on each other. If objects are determined to be independent no rules can be formulated however, if the objects are dependent a rule can potentially be formulated. Value $\mathcal{X}$ represents the transaction that contains the object $\mathcal{Y}$ and $\mathcal{N}$ is the number of transactions.

$$Confidence, c\ (\mathcal{X} \rightarrow \mathcal{Y}) = \frac{\sigma(\mathcal{X} \cup \mathcal{Y})}{\sigma(\mathcal{X})}$$

(Eq. 3.8)

$$Support, s\ (\mathcal{X} \rightarrow \mathcal{Y}) = \frac{\sigma(\mathcal{X} \cup \mathcal{Y})}{(\mathcal{N})}$$

(Eq. 3.9)

$$lift, l\ (\mathcal{X} \rightarrow \mathcal{Y}) = \frac{\sigma(\mathcal{X} \cup \mathcal{Y})}{\sigma(\mathcal{X}) \times \sigma(\mathcal{Y})}$$

(Eq. 3.10)

The efficiency of extracting the patterns by the machine learning algorithms will be measured using the *R* package tictoc in milliseconds. The tictoc is an *R* package used to measure the computational time taken for an algorithm over multiple steps to be completed. *R* was used to iterate through the application of the machine learning algorithms to process the three full and their associated reduced datasets, 100 and 1,000 times, in order to calculate the mean time taken for the algorithms to process the datasets, to gain a precise efficiency measurement.

3.6 Equipment and Resources

3.6.1 Equipment

Details of the various equipment and resources used as part of this study are presented in this section. The hardware specifications used to conduct this study are presented in Table 3.3.

Table 3.3, Shows the hardware equipment, specification and a description of the task conducted on the device used in this study

Name	***Specification***	***Description***
Desktop Computer	**OS:** Ubuntu 14.04 LTS **RAM Memory:** 15.7Gib **Processor:** AMD FX(The Department of the Prime Minister and Cabinet)-8120 Eight-Core Processor x8 **Graphics:** Gallium 0.4 on NVE7	Experimental phase, training and testing selected algorithms on adversary datasets and recording results.
MacBook Pro Retina	**OS:** macOS Sierra 10.12.5 **RAM Memory:** 16GB **Processor:** 2.9 GHz Intel Core i5 **Number of Processors**: 1 **Total number of Cores:** 2 **Graphics:** Intel Iris Graphics 6100	Preliminary analysis, constructing experiments, evaluating and analysing results including documentation of results.

3.6.2 Resources

R-Studio, an open-source integrated development environment (IDE) for the statistical programming language *R*, was selected as the primary platform to execute the experiments. *R* has an assortment of machine learning algorithm packages including the package required for the chosen algorithms used in this study. The packages used for this study are outlined in Appendix A. *R* was used throughout the preliminary analysis phases along with *Microsoft Excel*, *Bash* and *Python 2.7*. Details of the additional software applications utilised in this study are presented in Table 3.4.

Table 3.4, Shows the software, version and a description of the task conducted using the application within this study

Software	***Version***	***Description***
R-Studio	1.0.135	An integrated development environment (IDE) for the statistical programming language *R*. Used for preliminary analysis and to execute the experiments.
Microsoft Excel for Mac	15.33	Spreadsheet application developed by Microsoft. Used within the pre-processing phase, in particular the data filtering step and the data transformation step.
Python 2.7	2.7.10	A programming language. Particularly used for Dataset two and Dataset three in the data understanding phase. As well as the data reduction step in the pre-processing phase.
Bash	3.2.57(L. Bilge, Dumitra, & #351)	A command language. Particularly used for Dataset two and Dataset three in the data understanding phase. As well as the data integration step in the pre-processing phase.

The honeypot datasets acquired for this study had been obtained from external sources. Dataset one and Dataset two had been obtained from Edith Cowan University's Security Research Institute (ECU SRI) while Dataset three has been acquired from an affiliate of ECU SRI. The data obtained by the external sources had been capturing adversary interactions for an extended period of time. These provided a longitudinal representation of adversary activities as well as capturing "real" adversary activities rather than generated attacks. Dataset one consists of data collected between 2012-07-23 and 2016-01-13 a period of 1270 days. Data in Dataset two was collected between 2017-05-06 and 2017-09-05 a period of 123 days. Dataset three consists of data collected data between 2012-10-27 and 2016-02-23 a period of 1215 days.

All three honeypot datasets had been acquired from SSH honeypots. Dataset one and Dataset three are acquired from Kippo SSH honeypots. Dataset two was acquired from a Cowrie SSH honeypot, the successor to the Kippo SSH honeypot. The Kippo and Cowrie SSH honeypots are medium interaction honeypots that feature a subset of the SSH service functionalities as exhibited by a 'genuine' system. A risk of obtaining data from external sources is the unknown suitability of the datasets for this study. Section 4.1 is the data understanding phase in which the data is examined and analysed in order to deem it suitable for this study.

3.7 Ethical Considerations

Ethical concerns had been taken into consideration for this study and there are no animal involvement nor any human interactions within this study. The data gathered by the honeypots cannot positively unequivocally identify a particular adversary. The recorded Internet Protocol (IP) address is only of the attacking IP and may not reflect the location of the adversary launching the attack. However, this study does not disclose the full attacking IP address, thus protecting the anonymity of the adversary, abiding by the limitations approved by Edith Cowan University's Ethics Approval board. This study was undertaken according to the approved ethical declaration.

4 Preliminary Analysis

The preliminary analysis chapter consists of the data understanding phase and the pre-processing phase. The project understanding phase is omitted from this chapter as the phase has already been presented in Section 3.4.1. The data understanding phase is the initial exploration and analysis of the datasets intended to be utilised for this study and is elaborated in Section 4.1. this is followed by the pre-processing phase, where the datasets deemed suitable for this study are prepared prior to the application of the selected machine learning algorithms, presented in Section 4.2.

4.1 Data Understanding

In the data understanding phase, the datasets intended to be utilised for this study are presented. An initial exploration and analysis were conducted to determine whether the datasets are suitable for this study. The procedure for the data understanding phase is outlined in Section 3.4.2. For this study, data has been acquired from medium interaction honeypots that emulate a SSH service. The data had been gathered by two Kippo SSH honeypots and a Cowrie SSH honeypot. SSH dataset one and SSH dataset two have been acquired from Edith Cowan University's Security Research Institute (ECU SRI) while SSH dataset three has been acquired from a source affiliated with the ECU SRI. The three datasets intended to be utilised for this study are presented in Table 4.1.

Table 4.1, Shows the three SSH datasets that will be utilised in this study

Datasets	***Type of SSH honeypot***	***Acquired from***	***Format***
Dataset one	Kippo SSH honeypot	ECU SRI	.csv
Dataset two	Cowrie SSH honeypot	ECU SRI	.log
Dataset three	Kippo SSH honeypot	An affiliate of ECU SRI	.log

As shown in Table 4.1, dataset one was acquired as a set of Comma Separated Values (CSV) files, while dataset two and dataset three had been acquired as a set of log files (.log). The log files had been acquired as a set of ttylog files, that display the recorded interactions between an adversary and the honeypot per unique session. In the proceeding sections, exploration and analysis of the three acquired datasets are detailed to determine the suitability of the datasets for this study.

4.1.1 Dataset One

Dataset one was acquired as a set of CSV files from ECU SRI, where data had been gathered from six Kippo SSH honeypots. The six honeypots had been placed in different geographical locations. The honeypots referred to as Bronx, Dugite and Goanna had been located in the Netherlands While the Bobtail, Magpie and Mopoke honeypots had been located in the United States of America. The procedure outlined in Section 3.6.2 was used to setup these six honeypots. The honeypots had been

configured to collect and store data tables, auth, client, input, sensor, session and ttylog as presented in Figure 4.1.

Figure 4.1 illustrates the attributes and the relationship between the six tables that form each honeypot dataset and in turn form dataset one. The session table is the central table with a relationship to the other five tables. The session table stores the unique session identification created when attempts have been made to gain unauthorised access to the honeypots. The auth table is associated with the session table and stores the credentials an adversary has used to attempt to gain unauthorised access to the honeypots. Once the adversary has successfully authenticated into a honeypot, the interactions the adversary has with a honeypot is stored in the input table. The sensor table stores the Internet Protocol (IP) address of the honeypot the adversary has attempted to gain access to, while the client table stores the version of the SSH client the adversary has utilised. The ttylog table stores the interaction between the adversary and the honeypot for a unique session. The data in the ttylog table can also be saved as a set of log files for each unique session.

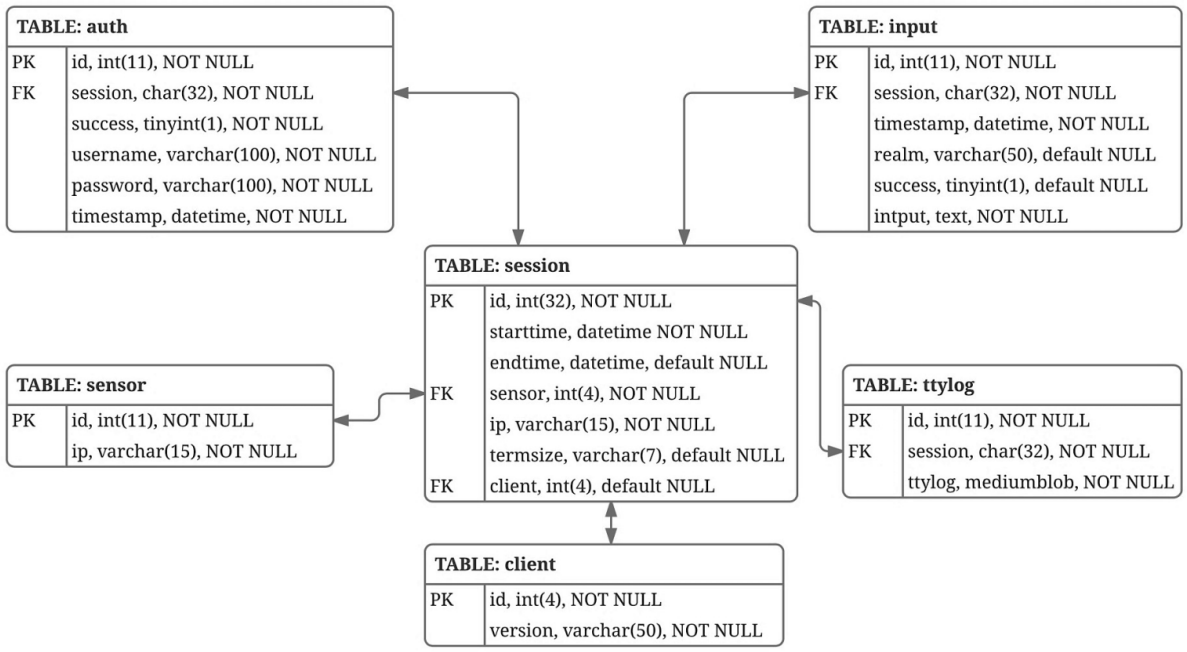


Figure 4.0.1, Illustrates the MySQL dataset structure for Kippo honeypot (Rabadia et al., 2017)

The input tables from each of the honeypots had been selected to be utilised for this study as the adversary interaction data per unique session is stored within the table and is the focus of this study. The attributes in the input table as shown in Figure 4.1 are, id, session, timestamp, realm success and input. The id attribute in the input table stores the unique identification value assigned to each recorded

input in order of occurrence. The session attributes correspond to the session the input occurred within. The date and time the input recorded are stored in the timestamp attribute. The realm attribute has a *NULL* value recorded for all samples in all six honeypots. The success attribute represents whether the input was successful based on a set of acceptable inputs. Adversarial interactions that have occurred between the honeypots are recorded within the input attribute. Table 4.2 shows the number of samples in each of the input tables of the six honeypots.

Table 4.2, Shows the number of samples in the input table of the six Kippo SSH honeypots combined to form SSH dataset one

Kippo SSH Honeypot	***Number of samples in the input table***
Bobtail	2,316
Bronx	1,476
Dugite	1,692
Goanna	1,600
Magpie	581
Mopoke	6,914
Total	**14,579**

The examination and analysis conducted have determined dataset one to be suitable for utilisation in this study in particular, the input tables of the six honeypots. The next phase for dataset one is the pre-processing procedure outlined in Section 4.3.1. In the following section, the examination and analysis for dataset two are elaborated.

4.1.2 Dataset Two

Dataset two has been acquired from ECU SRI as a set of log files. The log files contain ttylog session logs from a Cowrie SSH honeypot. In total 5,475 log files have been acquired. The ttylog files include the input commands of the adversary and the corresponding responses of the Cowrie SSH honeypot. However, the data firstly needed to be converted to a readable format to determine the suitability of dataset two for this study. A *Python 2.7* script called playlog.py (Michel Oosterhof, 2018b) was used to playback the ttylog file with the adversary interaction session from start to the end.

Bash was used to iterate the playlog.py *Python 2.7* script through the ttylog file with a grep command to extract only the adversary interactions. Thereafter the output was saved to a CSV file along with the associated log file name. The log files had been separated into six groups according to the size of the log files. The file sizes ranged from 24 bytes to 34 kilobytes. The log files have been separated to monitor the convergence process to maintain the integrity of the CSV files produced. Allowing for any potential problems to be identified and ratified. The CSV file format had been selected as the file format can be accessed across different applications, such as *R*, *WEKA* and *Sublime*.

Analysing the six CSV grouped files had shown no adversary interaction data was found in file sizes between 24 bytes and 48 bytes. Additional analysis found the log files are created once an adversary has successfully authenticated to the honeypot. An adversary can terminate the session without further interaction with the host but a log file is still created upon authentication. The log file size is dependent on the time taken for the adversary to terminate the session. Out of 5,475 ttylog session files, only 719 logs contained adversary interaction data across four CSV files. The initial exploration and analysis had shown dataset two was suitable to be utilised for this study as the dataset contains 719 sessions of adversary interaction data. In the pre-processing phase presented in Section 4.2.2, dataset two is massaged and prepared for the chosen machine learning algorithms to be applied.

4.1.3 Dataset Three

Dataset three was acquired as a set of ttylog files from an affiliate of ECU SRI. The ttylog files acquired as part of dataset three are similar to those in dataset two. Each log file contained the interaction between an adversary and the responses of the Kippo SSH honeypot, for a unique session. The same procedure applied to dataset two of converting the log files to a readable format was undertaken on dataset three.

The ttylog session files had been split into six files according to the size of the log. The sizes of the log files ranged from 85 bytes to 344 kilobytes. *Bash* was used to iterate the *playlog*.py (Michel Oosterhof, 2018b) *Python 2.7* script through the log files with a grep command to extract only the adversary interactions. Thereafter the output was saved to a CSV file along with the associated log file name. The CSV file format had been selected as the file format can be accessed across different applications, such as *R*, *WEKA* and *Sublime*. No adversary interaction data had been found in log files 85 bytes in size. Similar to the dataset two, a ttylog session log file is created upon correct authentication to the Kippo SSH honeypot, even when the adversary did not interact with the honeypot. This resulted in 318 sessions adversary interaction sessions from 4,995 log files. After exploration and analyse of the Kippo SSH ttylog session files the dataset was determined to be suitable to utilise in this study. Dataset three encompasses five files with a total of 318 unique adversary interaction sessions. In the pre-possessing phase in Section 4.2.3, dataset three is prepared to allow the selected machine learning algorithms to be applied.

4.1.4 Summary of the Data Understanding Phase

In the data understanding section, the three datasets shown in Table 4.1 have been deemed suitable for use in this study. The datasets have been acquired as either a set of CSV files or a set of ttylog session files. Dataset one had been acquired as a set of CSV files. Exploration of the dataset identified the data in the input table fulfils the requirements of this study. A total of 14,579 input commands had been stored within the input tables of the six Kippo SSH honeypots.

However, ttylog session files had been acquired for dataset two and dataset three. The ttylog files had been converted into a readable format and saved as a set of CSV files. Dataset two contained 719 sessions of adversary interaction data. While dataset three contained 318 sessions of adversary interaction data. Concluding the data understanding phase, the next phase is the pre-processing phase elaborated in Section 4.2.

4.2 Pre-Processing

Data pre-processing is a critical phase in the CRISP-DM model (Malley *et al.*, 2016; SPSS, 2000). In the pre-processing phase, the datasets are massaged and prepared to be applied to the selected machine learning algorithms. At the conclusion of the phase, the three datasets are organised into the same format to allow the selected machine learning algorithms to be tested on the datasets objectively. However, there are a combination of steps that can be taken in the pre-processing phase depending on the requirements of this study being undertaken and the data collected. Within the literature, there are variations of the pre-processing phase (Bramer, 2013; García, 2015; Hackeling, 2014). A pre-processing procedure with five steps has been developed for this study with the data reduction step also taking place in this phase as depicted in Figure 4.2. Figure 4.2 is an overview of the pre-processing procedure developed for the study with the flowchart illustrated Figure 3.4 showing the workflow. The pre-processed procedure developed for this study was first deployed on dataset one, followed by dataset two and then dataset three.

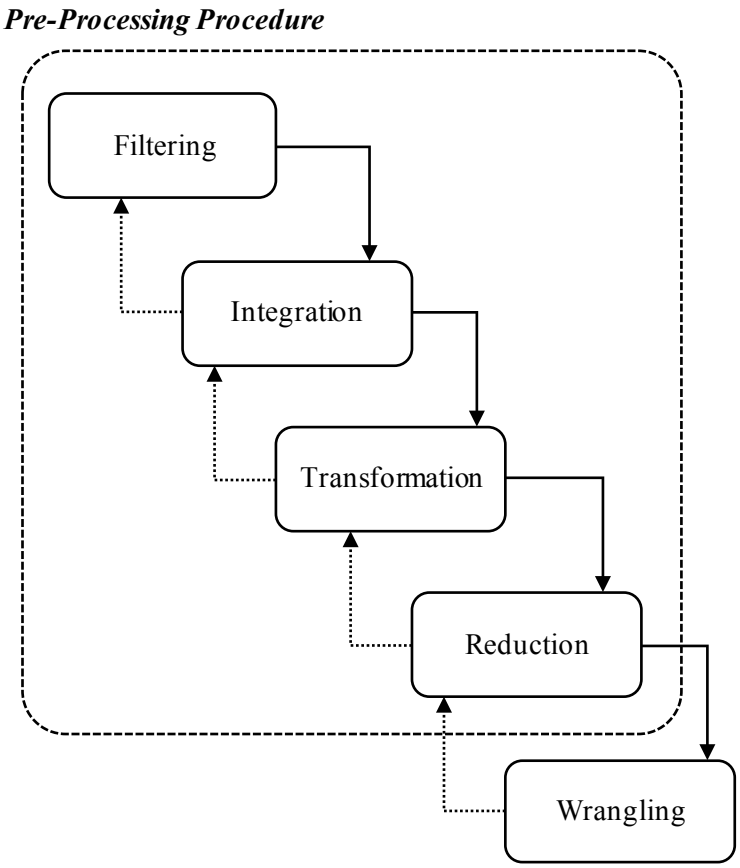


Figure 4.0.2, Depicts the five steps in the pre-processing procedure used in this study. The data wrangling step is outside the boundary as the step will occur prior to applying a chosen machine learning algorithm.

There are five steps in the pre-processing procedure that follow an iterative process: data filtering, data integration, data transformation, data reduction and data wrangling.

In the data filtering step data is sanitised removing any incomplete samples, redundant samples and other noisy data. Redundant data is unnecessary duplicates of data. It can be in the form of unnecessary duplicates of rows in a table or individual entries in a column. Noisy data can be corrupt data or problem data that is meaningless in the current format. Noisy data can cause incomplete samples and impact on the integrity of the dataset.

The data integration step is where tables from the same dataset or tables from different datasets are combined before relevant attributes are selected and extracted in preparation for the data transformations step.

The data transformation step involves transforming the data from the source to allow machine learning algorithms to be applied. The three datasets are duplicated before applying the reduction step to the duplicated dataset.

The data reduction step is to represent the data using an evenly distributed reduced dataset. Allowing for machine learning algorithms to be applied to the full and corresponding reduced dataset, testing the hypotheses in Section 3.2.

Prior to the application of the chosen machine learning algorithms to a dataset the final pre-processing step of data wrangling will take place. Hence the data wrangling step is outside of the pre-processing phase boundary and is discussed in Section 4.2.4 instead of within Sections 4.2.1 to 4.2.3. In the data wrangling step the data type is verified before the application of the machine learning algorithm. The data wrangling step will be discussed at the end of the chapter.

The five pre-processing steps follow an iterative process as previous steps may need to be conducted again if the objective of the current step is not achieved upon completion. The pre-processing phase depicted in Figure 4.2 had been applied to all three datasets, and are elaborated in the following sections.

4.2.1 Dataset One

Dataset one was deemed suitable for this study, as explained in the data understanding phase in Section 4.1.1 the dataset contained adversary interaction data. The initial exploration and analysis of dataset one indicated the input table from the six Kippo SSH honeypots are utilised since the data in the input tables contained adversary interaction commands. The following sections elaborate through the first four steps of the pre-processing procedure as depicted in Figure 4.2 applied to dataset one.

4.2.1.1 Data Filtering, Dataset One

The first step in the pre-processing phase is the data filtering step. Data filtering consists of removing redundant data, incomplete samples and other noisy data. Redundant data is unnecessary duplicates of data. It can be in the form of unnecessary duplicates of rows in a table or individual entries in a column. In dataset one there was no redundant data seen in the id columns of the input tables. The data in the session, timestamp and input columns of the input tables can have redundant data, as adversaries can interact with the honeypots more than once in a session. However, noisy data had been identified in the input column of the input tables causing incomplete samples in one of the six Kippo honeypot datasets. Noisy data can be corrupt data or problem data that is meaningless in the current format and can cause incomplete samples that can impact on the integrity of the data.

When the Mopoke honeypot dataset was loaded into *Microsoft Excel* and *R*, input strings leading with a dash '-' were taken as incomplete functions in both applications. Out of a total of 6,914 samples, only 17 samples had been affected, examples of these strings are: '-ls', '-sl', '-n' and '-profile'. Six sessions

originating from six different IPs had been identified that used a leading dash, four of which originated from Turkey, one from Romania and one from Western Australia. In order to clean the samples, the leading dash '-' was replaced with '(dash)' removing the noisy data and completing the samples.

4.2.1.2 Data Integration, Dataset One

In the data integration step, the six honeypot datasets are merged and attributes are selected to provide a holistic view of the data as well as resolving conflicting data. Data can be merged with other tables in the same format. For example, merging the input tables from the different honeypot datasets. Alternatively, relatable tables can be merged such as the input tables and session tables from the same honeypot dataset. Once the selected tables had been merged, the relevant attributes identified in the data understanding phase can be extracted.

With regards to dataset one, all six input tables had been combined together using the *Bash* command 'cat *.csv >> output.csv' and loaded into *R*. The combined input table had been viewed in *R* to verify the six input tables had been combined properly using the *Bash* command above and the integrity of the data had not been compromised. A total of 14,579 samples had been combined and 1,128 unique adversary's sessions had been identified. The input tables from the honeypot datasets had been combined as the requirements of this study focused on the adversary command chain per session not commands per IP address. As it is difficult to definitively determine only one adversary attacked a given honeypot using only one IP address. It is possible multiple adversaries could have utilised the same IP address to attack a given honeypot or one adversary could have used multiple IP addresses by utilising VPNs (Virtual Private Networks) to route through various locations. For this reason, the command chain per session was the focus as opposed to the command chain per IP address.

From the combined dataset attributes selection took place and relevant attributes to this study had been identified and extracted. To extract the input command per unique session: id, session, timestamp and input columns from the combined input dataset had been extracted, dropping the realm and success columns. The id attribute had been extracted as it shows the order of the adversary commands as they occur since the timestamp attribute only shows when commands had occurred to the nearest second. Both attributes are required for ordering the input commands per session. The session attribute is selected as one session can have multiple input commands. Finally, the input attribute is selected as the commands entered by adversaries are recorded in this column. The realm and success attributes had been dropped as the columns do not contain any data valuable to this study, the success of the commands entered by adversaries are out of scope.

4.2.1.3 Data Transformation, Dataset One

Data transformation is consistently transforming the source data format in preparation for the selected machine learning algorithms to be applied. The data transformation step consists of two components: transposing the data and tidying the data. Transposing the data involves converting the columns into rows and rows into columns. Data tidying is organising the data to show only one observation per row. The data transformation step allows for only one observation to be made between the variables and ensuring other relationships between multiple variables do not influence the outcome. For the combined input table, a typical transposition in *R* does not have the desired outcome and the dataset would be difficult to read, with 1,128 columns and four rows. It was necessary for the combined input table to be transposed during the transformation process.

The *R* package dplyr was used to only transpose the input column, ordering the elements based on the: id, session and timestamp. The resulting dataset combined all the input commands for each session into one column separated by a comma (,). The Text to Column function in *Microsoft Excel* was used to split the input column base on the comma ',' delimiter. Thereafter the dataset table was tidied by dropping the id, and timestamp columns to reduce redundant data as presented in Table 4.3. Table 4.3 shows the format of the combined input table after transposition had taken place where each record has the unique session_id followed by the commands utilised by the adversary in order.

From the newly transposed and tided input table the average adversary commands per session for dataset one was ~13 (12.92) commands in length, with the longest command chain 177 commands in length. To improve the performance of the selected machine learning algorithms, each unique command was assigned a unique key value in the format "comXX". For example replacing 'wget:http://127.0.0.1:8080/User2' with "com316". A total of 2,960 unique commands had been identified and extracted. *Python 2.7* was used to extract the unique commands and assign a unique key value to each command. The resulting file was then merged with the combined file from the data integration step then iterated back through the data transformation step. A sample of the resulting formatted, transposed and tided dataset is shown in Table 4.4, concluding the data transformation step.

Table 4.3, Shows a sample of the transposed and tidied dataset

session_id	***C1***	***C2***	***C3***	***C4***
0089a2d625ff11e589ef3787aa29d241	./32 &	chmod 0777 32	exit	wget http://127.0.0.1:8080/32
00c8b5522d4811e5a518757947861b68	./32 &	chmod 0777 32	wget http://127.0.0.1:8080/32	wget http://127.0.0.1:8080/32

Table 4.4, Shows a sample of the formatted, transposed and tidied dataset

session_id	*C1*	*C2*	*C3*	*C4*
0089a2d625ff11e589ef3787aa29d241	Com6	Com108	Com154	Com381
00c8b5522d4811e5a518757947861b68	Com6	Com108	Com381	NA

4.2.1.4 Data Reduction, Dataset One

The purpose of the data reduction step is to represent a full dataset using an evenly distributed reduced dataset. Prior to the application of the data reduction step, dataset one was duplicated and the data reduction step was applied only to the duplicated dataset. The focus of this study was to investigate whether a reduced dataset can be used to extract comparable patterns that are precise and the process of extraction is efficient compared to the related full dataset, as outlined in Section 3.2. Below the data reduction process for dataset one is elaborated and the combination and permutation of Full Dataset One (FD1) and the associated Reduced Dataset One (RD1) are calculated to highlight the variance in the volume of data.

To achieve a reduced dataset a clustering technique was developed, as the focus of this study was on the sequence of the commands in a chain as opposed to the sequence of individual commands entered. A total of 2,960 unique commands had been identified and extracted. *Python 2.7* was used to generalise all 2,960 commands according to the function of the command forming clusters. Thereafter each cluster was assigned a unique key value with the output saved to a file. For example: 'wget:http://127.0.0.1:8080/User2' was clustered to 'wget:http://ip:port/file' and assigned a unique key value of "eve316". Each unique key value is in the format "eveXX". The resulting file was then merged with the combined file from the data integration step then iterated back through the data transformation step. The number of unique commands had been reduced from 2,960 in FD1 to 406 in the associated RD1 a reduction of 86.284%.

Provided below are the combination and permutation values for FD1 and RD1, to quantify the variance in the volume of data to show the reduction in the size of the reduced dataset. To calculate the possible x number of combinations occurring from a set of n objects, the combination equation (Eq 4.1) was utilised. For FD1 n was assigned 2,960 as it is the number of unique commands in the dataset. x was assigned 13, as it is the average number of adversary commands per chain. Resulting in a combination value of ~ 2.094 quintrigintillion (2.094360976 E+35) for FD1.

As for the associated RD1, n was assigned 405 as it is the number of clustered commands in the dataset. The average adversary number of commands per chain had not altered, hence x was assigned 13. The result was a combination value of ~1.043 quattuorvigintillion (1.042603223E+24) for RD1.

$$Combination = \frac{n!}{x!(n-x)!}$$

(Eq 4.1)

Permutation is used to calculate the number of *n* objects in a set arranged in order of *x* per group. The permutation equation (Eq 4.2) utilised the same assigned values for *n* and *x* as the combination calculations for FD1 and RD1. The permutation value for FD1 was ~1.304 quinquardragintillion (1.304162936E+45) whereas the permutation value for RD1 was ~6.492 tretrigintillion (6.492311959E+33).

$$Permutation = \frac{n!}{(n-x)!}$$

(Eq 4.2)

The combination and permutation values for FD1 and RD1 show the variances in the volume of data. In the experimental phase elaborated in Section 3.5.4, the selected machine learning algorithms are applied to FD1 and RD1. Testing whether RD1 can be used to extract comparable patterns that are precise and the process of extraction is efficient compared to the related full dataset.

The four steps of the pre-processing phase had been completed for dataset one resulting in a full and reduced representation of dataset one. The final step of data wrangling is conducted prior to the application of the chosen machine learning algorithms. In the following sections, the first four pre-processing steps are applied to dataset two.

4.2.2 Dataset Two

The pre-processing phases illustrated in Figure 4.2 are also applied to dataset two however, the data filtering step differs from the steps applied to dataset one in Section 4.2.1. As dataset two had been acquired as a set of ttylog session files. Nevertheless, at the completion of the pre-processing phase, dataset two would be organised to resemble the processed FD1 and RD1.

In the previous data understanding phase in Section 4.1.2, the log files had been grouped according to the size of the file. The adversary interaction data from the ttylog files along with the associated log file name had been extracted and collated into a CSV file for each group of files using *Bash*. A total of 719 adversary interaction sessions had been recorded across four files for dataset two.

In the following data filtering step the CSV files have been sanitised removing noisy data. In the data integration step the four CSV files are merged allowing for the data to be transposed and transformed in the data transformation step. This is followed by producing a reduced dataset of the Full Dataset Two (FD2) in the data reduction step. Finally, in the data wrangling step the data types are converted to the

required type according to the machine learning algorithm to be applied. The data wrangling step had been conducted prior to the application of the chosen machine learning algorithm.

4.2.2.1 Data Filtering, Dataset Two

Within the data filtering step, the dataset is sanitised, removing all redundant data, incomplete samples and other noisy data. Unlike the data filtering step of dataset one in Section 4.2.1.1, there are three stages to the data filtering step. The stages are illustrated in Figure 4.5 shows, the four CSV files of the dataset two had noisy data as seen in stage one, a single column containing the log name and the shell prompt command along with the adversary interaction commands. At the completion of the data filtering step, the four CSV files resembled stage 3 of Figure 4.5. The log name and adversary input commands had been separated into different columns and the shell prompt data was removed because it was noisy data.

To get the four CSV files from the stage one format to the stage two format shown in Figure 4.5, the Text to Column function in *Microsoft Excel* was used to separate the column by the hash '#' delimiter. The adversary interaction commands had been separated from the shell prompt by splitting the column based on the hash '#' delimiter. Any data after the hash '#' delimiter was the adversary interaction data.

Thereafter the shell prompt data was removed and the logid was assigned to the corresponding adversary interaction data. Additionally, an id was assigned to each of the samples to ensure the order of the input data would be maintained. Converting the four CSV files from stage two to the format shown in stage three of Figure 4.5 concluding the data filtering step for dataset two. An observation was made regarding one of the four CSV files, out of 719 unique adversary interaction sessions, 675 of these unique sessions had the same sequence of the input commands. Dataset two predominantly consisted of duplicate data suggesting a script was utilised to interact with the honeypot. The processing step is the data integration step for dataset two.

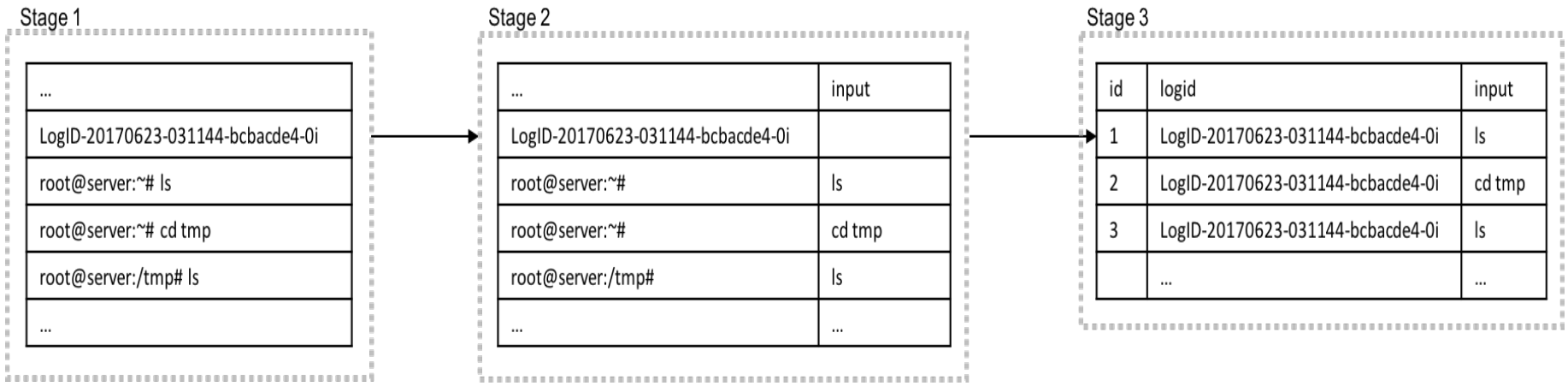


Figure 4.0.3, Depicts the three stages of data massaging in the data filtering step of the pre-processing procedure

4.2.2.2 Data Integration, Dataset Two

Within the data integration step, data is merged and attributes are selected to provide a holistic view of the dataset. For dataset two, a decision was taken to merge all four CSV files, despite one of the CSV files containing a possible script. This study focused on investigating whether a reduced dataset can be used to extract comparable patterns that are precise and the process of extraction is efficient compared to the related full dataset, as outlined in Section 3.2. Hence, a dataset that comprised predominantly of a possible script would not affect the focus of this study.

The same *Bash* command 'cat *.csv >> output.csv' used in the data integration step for dataset one was used to merge the four CSV files for dataset two to be loaded into *R*. In total dataset two comprised of 11,015 adversary interaction commands throughout 719 unique sessions. Attribute selection for dataset two unlike dataset one was not required as the three attributes: id, Logid and input are required for this study, concluding the data integration step.

4.2.2.3 Data Transformation, Dataset Two

Data transformation is consistently transforming the source data format in preparation for the selected machine learning algorithms to be applied. A typical transformation results in a data frame with 719 columns and three rows and was not the desired outcome. The *R* package dplyr used to transpose dataset one was also applied to transpose dataset two. Unlike dataset one, the timestamp attribute was absent from dataset two therefore the id attribute was used to order the input attribute for each session.

The resulting dataset combined all the commands for each session into one column separated by a comma (,). The Text to Column function in *Microsoft Excel* was used to split the input column base on the comma *','* delimiter. Thereafter, the dataset was tidied by dropping the id column. At this point dataset two was presented in the format shown in Table 4.3, each unique Logid value was followed by the order the sequences of commands utilised by the adversary. The recently transposed and tided dataset revealed the average command chain to be ~15 (15.312) commands in length, with the longest command chain at 45 commands in length. Similar to dataset one each unique command was assigned a unique key value in the format "comXX". For example, replacing 'wget:http://127.0.0.1:8080/User2' with "com316", in order to improve the performance of the selected machine learning algorithms. A total of 81 unique adversary commands had been identified and extracted. *Python 2.7* was used to extract the unique commands and assign a unique key value to each unique adversary command. The subsequent file was then merged with the combined file from the data integration step then iterated back through the data transformation step. A sample of the resulting formatted, transposed and tided dataset is shown in Table 4.4, concluding the data transformation step for dataset two.

4.2.2.4 Data Reduction, Dataset Two

The purpose of the data reduction step is to represent FD2 using an evenly distributed reduced dataset. Similar to dataset one prior to the application of the data reduction step dataset two was duplicated, the data reduction step was applied only to the duplicated dataset. Since the focus of this study was to investigate whether a reduced dataset can be used to extract comparable patterns that are precise and the process of extraction is efficient compared to the related full dataset, as outlined in Section 3.2. Below the data reduction process for dataset two is elaborated and the combination and permutation of FD2 and associated Reduced Dataset Two (RD2) are calculated to highlight the variance in the volume of data.

The clustering technique deployed on dataset one was utilised to reduce dataset two as the focus of this study was on the sequence of the commands in a chain as opposed to the sequence of individual commands entered. A total of 81 unique adversary commands had been identified and extracted. *Python 2.7* was used to generalise all 81 commands according to the function of the command forming clusters. Thereafter each cluster was assigned a unique key value with the output saved to a file. For example: 'wget:http://127.0.0.1:8080/User2' was clustered to 'wget:http://ip:port/file' and assigned a unique key value of "eve316". Each unique key value is in the format "eveXX". The resulting file was then merged with the combined file from the data integration step then iterated back through the data transformation step. The number of unique commands had been reduced from 81 in FD2 to 45 for RD2, a reduction of 44.444%.

The combination and permutation values have been calculated to quantify variance in the volume of data in FD2 and RD2. The combination equation shown in Eq 4.1 is used to calculate the possible x number of combinations that can occur from a set of n objects. Eq 4.2 is the equation used to calculate the permutation value. Permutation is used to calculate the number of n objects arranged in order of x per group. To calculate the combination and permutation values of FD2, n was assigned 81 as it is the number of unique commands in dataset two. While x was assigned 15 as it is the average number of adversary commands per chain. FD2 has a combination value of ~8.144 quindecillion (8.144022047E+15) and a permutation value of ~1.065 octovigintillion (1.064972888E+28).

After the data reduction step was applied, by clustering the commands based on the function, n was assigned 46 as it is the number of unique adversary commands in RD2. While x was assigned 15 as the average adversary number of commands per chain had not altered. The commination value of ~3.449 undecillion (3.448674255E+11) was calculated for RD2 and a permutation value of ~4.510 trevigintillion (4.509742927E+23).

The combination and permutation values for FD2 and RD2 show the variation in the volume of data. The four steps of the pre-processing phase had been completed for dataset two, resulting in a full and reduced representation of dataset two. The final step of data wrangling is conducted prior to the application of the chosen machine learning algorithms. In the experimental phase elaborated in Section 3.5.4, the selected machine learning algorithms are applied to FD2 and RD2. Testing whether a reduced dataset can be used to extract comparable patterns that are precise as well as evaluating the process of extraction for efficiency compared to that of the related full dataset. In the following sections, the first four pre-processing steps are applied to dataset three.

4.2.3 Dataset Three

For dataset three to resemble the processed dataset one and dataset two along with their associated reduced datasets, the pre-processing procedure illustrated in Figure 4.2 was applied to dataset three. The data filtering step was similar to that of dataset two as both datasets had been acquired as a set of ttylog session files.

In Section 4.2.3 the data understanding phase for dataset three was conducted. The initial exploration and analysis of dataset three grouped the log files according to size. The adversary interaction data from the ttylog files along with the associated log file name then collated into a single CSV file for each group of files using *Bash*. A total of 318 adversary interaction sessions had been recorded across five files for dataset three.

As per Figure 4.2, the pre-processing procedure applied to dataset three is elaborated in the following sections. However, the data wrangling step had been conducted before the application of the machine learning algorithms to the datasets, ensuring the data type is appropriate for the selected algorithm. The data wrangling step is discussed at the end of the chapter. The following sections elaborate the first four steps of the pre-processing procedure applied to dataset three.

4.2.3.1 Data Filtering, Dataset Three

In the data filtering step, the dataset is sanitised, removing redundant data, incomplete samples and other noisy data. Dataset three had two forms of noisy data. The first form of noisy data was the presence of control characters. Additionally, like dataset two, the shell prompt data was seen along with the adversary interaction data in a single column as shown in stage 1 of Figure 4.5.

Control characters are non-printable ASCII characters. Some applications such as the command terminal can read the control characters as intended. Whereas other applications such as *R* have issues interpreting the control characters due to the encoding of the file (R Studio, 2015). These issues include

inserting new incomplete samples, line breaks and the merging of cells. For these reasons, control characters can impact the integrity of a dataset. For example, the control character '\b' indicates a backspace, it will erase or overwrite the last character causing incomplete samples. The control character '\t' will add spacing to the right while '\n', will insert a new line, causing a line break. For these reasons the samples containing control characters had been removed. A text editor called *Sublime* was used to find and remove the regular expression 'ESC' and '[**' that represent the control characters in the encoding of the file. A total of 2,424 control characters had been identified and removed.

The second form of noisy data seen was the shell command prompt along with adversary interaction data. To get the five CSV files from the stage one format to the stage two format as shown in Figure 4.5 the Text to Column function in *Microsoft Excel* was utilised to separate the column by the hash '#' delimiter. The hash '#' delimiter was the last character of the command prompt, thereafter the data was adversary interaction data as shown in stage two of Figure 4.5.

In order to get the CSV files from stage two to stage three the shell prompt data was removed and the appropriate logid was assigned to the corresponding adversary interaction data. Additionally, an id was assigned to each sample to ensure the order of the input data would be maintained. Upon completing the data filtering step a total of 318 unique adversary interaction sessions had been recorded for dataset three.

4.2.3.2 Data Integration, Dataset Three

Similar datasets or tables are merged and attributes are selected to provide a holistic view of the data in the data integration step. The data integration step for dataset three involved merging the five CSV files. The selection of attributes for dataset three was irrelevant as the, id, logid and input attributes are already seen in the dataset.

The same *Bash* commands 'cat *.csv >> output.csv' used to merge data in the previous datasets was utilised to merge the five CSV files of dataset three. The merged dataset had then been loaded into *R* to verify the dataset had been merged correctly and the integrity of the data was not impacted. A total of 2,314 adversary interaction commands spread over 318 unique sessions had been recorded for dataset three.

4.2.3.3 Data Transformation, Dataset Three

In the data transformation step, the source data format is consistently transformed in preparation for machine learning algorithms to be applied. A typical transformation in *R* would result in a dataset with

318 columns and three rows and where not the desired outcome. The *R* package dplyr was used to transpose the previously processed datasets was used to transpose dataset three as well. Similar to dataset two, the timestamp attribute was absent therefore the id attribute was used to sort the input commands in order of occurrence for each session.

The resulting dataset combined all commands for each session into one column separated by a comma (,). The Text to Column function in *Microsoft Excel* was used to split the input column based on the ',' delimiter. Next, the dataset was tidied by dropping the id column. Dataset three presented in the same format as shown in Table 4.3, each unique Logid value was followed by the order the sequences of commands utilised by the adversary. The recently transposed and tided dataset three revealed the average command chain to be ~7 (7.28) commands in length with the longest command chain at 220 commands in length. Similar to dataset two each unique command was assigned a unique key value in the format "comXX". For example, replacing 'wget:http://127.0.0.1:8080/User2' with "com316", in order to improve the performance of the selected machine learning algorithms. A total of 748 unique adversary commands had been identified and extracted. *Python 2.7* was used to extract the unique commands and assign a unique key value to each unique adversary command. The subsequent file was then merged with the combined file from the data integration step then iterated back through the data transformation step. A sample of the resulting formatted, transposed and tided dataset is shown in Table 4.4, concluding the data transformation step for dataset three.

4.2.3.4 Data Reduction, Dataset Three

The purpose of the data reduction step is to represent a full dataset using an evenly distributed reduced dataset. Prior to the application of the data reduction step, dataset three was duplicated and the data reduction step was applied only to the duplicated dataset. The focus of this study was to investigate whether a reduced dataset can be used to extract comparable patterns that are precise and the process of extraction is efficient compared to the related full dataset, as outlined in Section 3.2. Below the data reduction process for dataset three is elaborated and the combination and permutation of the Full Dataset Three (FD3) and Reduced Dataset Three (RD3) are calculated to highlight the variance in the volume of data. The data reduction step was the same for all three datasets.

To achieve a reduced dataset the developed clustering technique was deployed, as the focus of this study was on the sequence of the commands in a chain as opposed to the sequence of individual commands entered. A total of 748 unique commands had been identified and extracted. *Python 2.7* was used to generalise all 748 commands according to the function of the command forming clusters. Thereafter each cluster was assigned a unique key value with the output saved to a file. For example: 'wget:http://127.0.0.1:8080/User2' was clustered to 'wget:http://ip:port/file' and assigned a unique

key value of "eve316". Each unique key value is in the format "eveXX". The subsequent file was then merged with the combined file from the data integration step then iterated back through the data transformation step. The number of unique adversary commands had been reduced from 748 in FD3 to 338 for RD3, a reduction of 54.813%.

To quantify the reduction of dataset for FD3 and RD3, the combination and permutation of the datasets were calculated. Combination Eq 4.1 was used to calculate the possible *x* number of combinations occurring from a set of *n* objects. Whereas Eq 4.2 is used to calculate the permutation value. Permutation is used to calculate the number of *n* objects arranged in order of *x* per group. For FD3 *n* was assigned 748 and *x* was assigned 7. The combination value for FD3 was ~2.527 sexdecillion (2.527274515E+16) while the permutation value was ~1.274 vigintillion (1.273746355E+20). RD3 was assigned an *n* value of 338 as it is the number of clustered commands in the dataset while the average adversary number of commands per chain had not altered, hence *x* was assigned 7. The combination value for RD3 was ~9.394 tredecillion (9.393532302E+13) and a permutation value of ~4.734 septendecillion (4.73434028E+17). The combination and permutation values for FD3 and show variances in the volume of data.

All three datasets have been organised into consistent formats and the associated reduced datasets have been generated. In the experimental phase, elaborated in Section 3.5.4, the selected machine learning algorithms are applied to the full datasets and associated reduced datasets. Testing whether a reduced dataset can be used to extract comparable patterns that are precise and the process of extraction is efficient compared to the related full dataset. The final pre-processing step of data wrangling step was applied to the datasets prior to the application of the machine learning algorithms. However, as the data wrangling step is part of the pre-processing procedure it is elaborated below.

4.2.4 Data Wrangling

The final step in the pre-processing phase is the data wrangling step although it was applied to the three full datasets and associated reduced datasets prior to the use of the selected machine learning algorithms. It is still part of the pre-processing procedure as outlined in Figure 4.2 as such is discussed below.

In the data wrangling step a dataset is formatted to allow for the improved performance of the machine learning algorithms to process the data. Each machine learning technique has a particular data type, such as nominal for the association rule class of techniques however, the FP-growth association rule algorithm requires a binary data type. For this reason the data wrangling step has been conducted prior to the application of the selected machine learning algorithm. Additionally, the data wrangle step includes replacing empty cells with 'NA' to indicate the cell has no assigned value.

Data is classified into different classes called data types. Within the domain of machine learning there are four main data type classes (Hackeling, 2014):

- Nominal: Data is categorised into sets, with the user required to select an element from the given set. An example of nominal data is eye colour (blue, brown, green, hazel).
- Ordinal: The ordinal data type allows for data to be scaled and compared. An example is student exam ranking, 1st, 2nd, 3rd etc.
- Interval: Observations are made using the distances between data points. With interval data type, a value of zero can be assigned. For example: measuring temperature in degree Celsius.
- Ratio: Ratio is similar to the interval data type, in allowing for observations to be made between data points. However, unlike the interval data type, ratio data types cannot be assigned a value of zero. For example, height or weight.

The data type class can vary between applications. *WEKA*, classes data into four different data types: nominal, numeric, string and dates. The nominal data type categories data into sets (a, b, c.) Numerical data type only consists of numbers (0-9) and is similar to the interval data type. Whereas the string data type can be alphanumeric (alphabet and numeric values) and have special characters. It is comparable to the ratio data type. The date data type uses the default ISO-8601 date time format (yyyy-MM-dd‘T’HH:mm:ss) and is similar to the ordinal data type (Ian Witten, Eibe Frank, Mark Hall, & Pal, 2016).

R has five data type categories: logical, numerical, integer, complex and characters. Logical data types are Boolean values (true, false and yes, no). Numerical data types are real and decimal point numbers (0.00-9.99). Integer data type is a whole number (0-9) and complex data types are unknown values and equations (5x + 3 = y). Character data types are simpler to the string data type in *WEKA*, consisting of alphanumeric (alphabet and numeric values) and special characters.

The data type is also determined by the machine learning technique and algorithm selected. For example, the association rule class of machine learning techniques requires data to be in the nominal data type in *WEKA*, and the character data type with a factor data structure in *R*. The classification class of machine learning techniques commonly use numeric or string format in *WEKA* and character format in *R*.

4.3 Summary of the Preliminary Analysis

Section 4.1 data understanding phase is an initial exploration and analysis of the three acquired datasets in order to determine the suitability of use within this study. Once the three datasets had been deemed suitable for this study, the pre-processing phase was applied. The pre-processing procedure developed

consisted of five steps the data filtering, data integration, data transformation, data reduction and data wrangling steps. The aim of the pre-processing phase in Section 4.2 was to prepare the datasets to be applied to the selected machine algorithms, as well as organising the three datasets in the same format. Additionally, producing a reduced version of the full dataset to test whether a reduced dataset can be used to extract comparable patterns that are precise as well as evaluating the process of extraction for efficiency compared to that of the related full dataset. After the first four steps of the pre-processing procedure had been applied resulting in three full and their associated reduced datasets. It was observed FD2 and RD2 predominantly consisted of duplicated sequences of commands. While FD3 and RD3 predominantly consisted of distinct sequences of commands. The final data wrangling step was applied prior to the application of each machine learning algorithm.

5 Results

In Chapter five the results obtained from the experiments conducted on the reduced datasets and their respective full datasets are presented. The experiments had been developed to test the hypotheses of this study, as outlined in Section 3.2. Section 5.1 presents the results from applying the Naïve Bayes and Markov chain probabilistic classification algorithms to the datasets. In Section 5.2 the results from applying the Apriori and Eclat association rule mining algorithms to the datasets are presented. The aggregated results obtained from Section 5.1 and Section 5.2 are presented in Section 5.3, along with the corresponding hypotheses tested.

5.1 Probabilistic Classification Algorithms

Probabilistic classification algorithms are machine learning algorithms that commonly utilise supervised learning techniques to extract patterns from a given dataset. Probabilistic classification algorithms use classified features to assess the probability of an instance occurring based on observations made within the given dataset. The probabilistic classification algorithms that had been selected for this study are the Naïve Bayes and Markov chain algorithms. The hypotheses that are tested in this section are *H1*, *H3* and *H5*. The probabilistic classification algorithms are applied to the datasets to access whether, additional class patterns can be extracted from the reduced datasets compared to those class patterns extracted from their respective full datasets (corresponding to *H1*). The precision of the patterns extracted from the datasets are also compared, to assess whether more precise patterns can be extracted from the reduced datasets compared to their respective full datasets (corresponding to *H3*). Along with experiments conducted to assess the efficiency of the algorithms to process the reduced datasets compared to their respective full datasets (corresponding to *H5*). Elaborated below are the sets of experimental results conducted using the Naïve Bayes algorithm shown in Section 5.1.1, followed by the experimental results conducted using the Markov chain algorithm presented in Section 5.1.2.

5.1.1 Naïve Bayes Algorithm

The experiments conducted in this section apply the Naïve Bayes algorithm to the three reduced datasets and their respective full datasets to test the *H1*, *H3* and *H5* hypotheses outlined in Section 3.2. Prior to applying the Naïve Bayes algorithm to the datasets, the data type was verified to ensure the data was presented as nominal data type (a factor data type in *R*). As well as ensuring any empty cells are represented using *NA* (not applicable) to indicate there is no data within the cell to be processed, whereas an empty cell indicates data is missing. Concluding the pre-processing phase elaborated in Section 4.2.

A train and test model was implemented to evaluate the performance of the trained Naïve Bayes classifier. The *R* package `naivebayes` implementation of the Naïve Bayes algorithm was used in this

study. The Naïve Bayes algorithm experiments began by randomly shuffling the datasets using a seed value of 123, before splitting up the shuffled dataset into a 70% training set and 30% testing set. The training feature chosen for each dataset varied depending on the average sequence of commands for the given datasets. The length of the commands within the reduced datasets had not been altered by the clustering process consequently, both the full datasets and their associated reduced datasets had the same average sequence of commands. Once the classifier was trained based on the chosen feature, it was applied to the test set (30% of the given dataset). A confusion matrix shown in Table 3.2, was used to evaluate the performance of the trained Naïve Bayes classifier to classify instances in the test dataset.

A confusion matrix shows the number of correct and incorrect instances classified by the trained classifier. The trained classifier uses patterns within the given dataset to classify instances. To evaluate whether additional class patterns can be extracted from the reduced datasets compared to those class patterns extracted from their respective full datasets, the True Positive (TP) rate was compared as it is the rate of correctly classified instances (responding to *H1*).

The confusion matrix is also used to calculate the precision of the trained classifier to classify instances. The precision (Eq. 3.1), accuracy (Eq. 3.2), sensitivity or recall (Eq. 3.3), F1 score (Eq. 3.4) and error rate (Eq. 3.5) for reduced datasets and their respective full datasets are calculated and compared (responding to *H3*). Further explanation is provided in Section 3.5.

The processing time in seconds(s) for the Naïve Bayes algorithm to process the reduced datasets and their respective full datasets had been ascertained, by iterating the training and testing process 100 and 1,000 times. The process was iterated 100 times as opposed to 1,000 times, as evaluating the precision for the algorithm for 1,000 iterations would be computationally intensive. Thereafter, the mean processing time was calculated once the outliers had been removed, to achieve a precise efficiency measurement to be assessed (responding to *H5*). The sections below shows the results from applying the Naïve Bayes algorithm to the three reduced datasets and their respective full datasets, to test the hypotheses associated with this study.

5.1.1.1 Naïve Bayes algorithm applied to Datasets One

In this section, the Naïve Bayes algorithm is applied to Reduced Dataset One (RD1) and the respective Full Dataset One (FD1). Both datasets contain a total of 1,128 sessions of adversary interactions. The longest sequence of adversary commands was 177 commands in length. The 13th command in the sequence was computed to train the Naïve Bayes classifier as it is the average length of a command sequence for both datasets. Both datasets had been randomly shuffled using a seed value of 123, then

split into a 70% (790 sessions) training set and 30% (338 sessions) a testing set. The results obtained from the experiments conducted to test hypotheses *H1*, *H3*, *H5* are presented below.

A classification algorithm mines for patterns within a dataset based on the chosen feature. Inferring true patterns are seen if the classification algorithm can predict instances within a given dataset, irrespective of whether the instances are correctly classified. The trained classifier was applied to the test set of both datasets, to verify whether the instances or commands had been classified correctly by the trained classifier. For purpose of this study, the TP rate was used to determine if additional class patterns can be extracted from the RD1 compared to those class patterns extracted from FD1 (responding to *H1*). The confusion matrix was used to calculate the TP rates. The TP rate for FD1 was 71.623% (53/74) while the TP rate for RD1 was 77.027% (57/74). RD1 had a greater TP rate suggesting additional patterns can be extracted from RD1 compared to those extracted from FD1.

The precision of the classified instances was also calculated using the confusion matrix. The values in Table 5.1 are on a scale between 0 and 1, where 1 is the greatest value given. Precision is used to measure the correctness of the classification algorithm when classifying data. The precision measurement for the Naïve Bayes classifier of FD1 was greater at 0.660 compared to that of RD1 recorded at 0.646, a difference of 0.014. Sensitivity is also known as the recall (true positive rate), represents the number of objects correctly classified as either a true positive or true negative. Similar to the precision value, the sensitivity value for RD1 was greater, recorded at 0.461. Compared to the 0.429 sensitivity value for FD1. The F1 score measures the predictive probability of the classification produced by calculating a balanced mean between the precision and sensitivity. The F1 score was also greater for FD1 compared to the associated RD1, with a difference of 0.032. With the F1 scores for FD1 recorded at 0.543 and 0.515 for RD1.

However, the accuracy rate is used to measure the probability of the truly classified data was greater for RD1 at 0.770 compared to the 0.716 accuracy rate recorded for FD1, a difference of 0.054. The error rate measures the error of incorrect classified data with a lower error rate desirable, FD1 recorded an error rate of 0.284 while RD1 recorded a lower error rate of 0.230. From the results presented in Table 5.1, the accuracy and error rate are greater in RD1. Though the precision, sensitivity and F1 scores are greater for FD1 (responding to *H2*).

Table 5.1, Shows the precision, accuracy, sensitivity, F1 score and error rate of the classified instances using the trained Naïve Bayes algorithm for RD1 and FD1. Calculated using the confusion matrices for each dataset, the measurements are on a scale between 0 and 1, where 1 is the highest.

Naïve Bayes	***FD1***	***RD1***
Precision	0.660	0.646
Accuracy	0.716	0.770
Sensitivity (Recall)	0.461	0.429
F1 Score	0.543	0.515
Error Rate	0.284	0.230

To evaluate the efficiency of the Naïve Bayes algorithm to process RD1 and FD1, the training and testing stages had been iterated 100 and 1,000 times and the processing time for each iteration recorded in seconds(s) (responding to *H5*). The black lines seen in Figure 5.1 and Figure 5.2, represent the mean processing time once the outliers had been removed. The Figure 5.1, shows the processing time of the Naïve Bayes algorithm applied to the datasets iterated 100 times. The Naïve Bayes algorithm processed RD1 more efficiently compared to FD1. The highest processing time recorded for Figure 5.1 was recorded for FD1 at 2.340s. While the highest recorded processing time for RD1 was recorded at 2.023s. The lowest recorded processing time for Figure 5.1 was for RD1 was 1.771s. With the lowest processing time for FD1 was recorded at 2.070s.

A total of 11 outliers had been identified and removed before a mean processing time for FD1 was calculated at 2.103s. Whereas the mean processing time for the RD1 was recorded at 1.805s after removing five outliers, there is a difference of 0.299s between the mean processing time for both datasets. Suggesting the Naïve Bayes algorithm is more efficient at processing RD1 compared to FD1.

Figure 5.2 illustrates the processing time for RD1 and FD1 by the Naïve Bayes algorithm with the process iterated 1,000 times. As seen in Figure 5.1, the algorithm was efficient in processing RD1 compare to FD1 and is also observed in Figure 5.2. The processing time of FD1 ranged between 2.042s and 2.369s. Whereas the processing time for RD1 was ranged between 1.763s and 1.824s.

A total of 83 outliers had been removed before calculating the mean processing time of 2.097s for FD1. The mean processing time for RD1 was calculated at 1.805s after 49 outliers had been removed. The mean processing time for both datasets are represented by the black lines illustrated in Figure 5.2. There is a difference of 0.292s between the mean processing time for FD1 and RD1. The aggregated results are presented in Section 5.3 along with the corresponding hypotheses tested. In the following section the results from applying the Naïve Bayes algorithm to Reduced Dataset Two (RD2) and the respective Full Dataset Two (FD2).

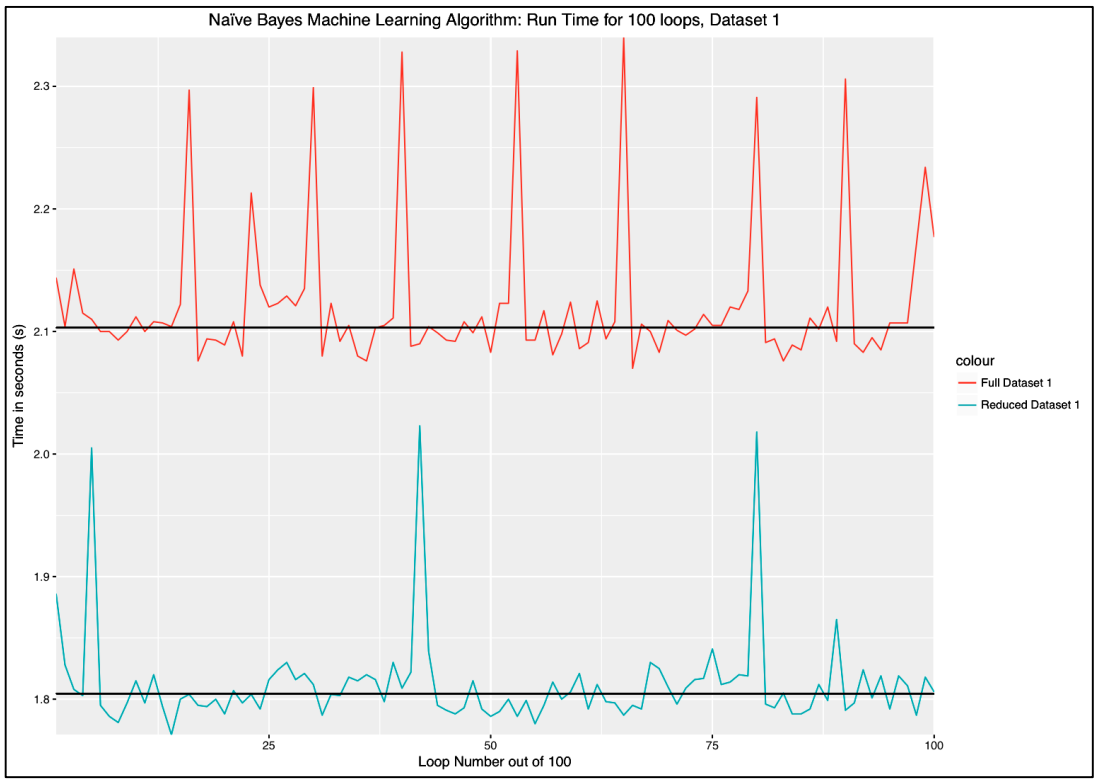


Figure 5.0.1, Illustrates the processing time in seconds(s) for the Naïve Bayes algorithm to process RD1 and FD1. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

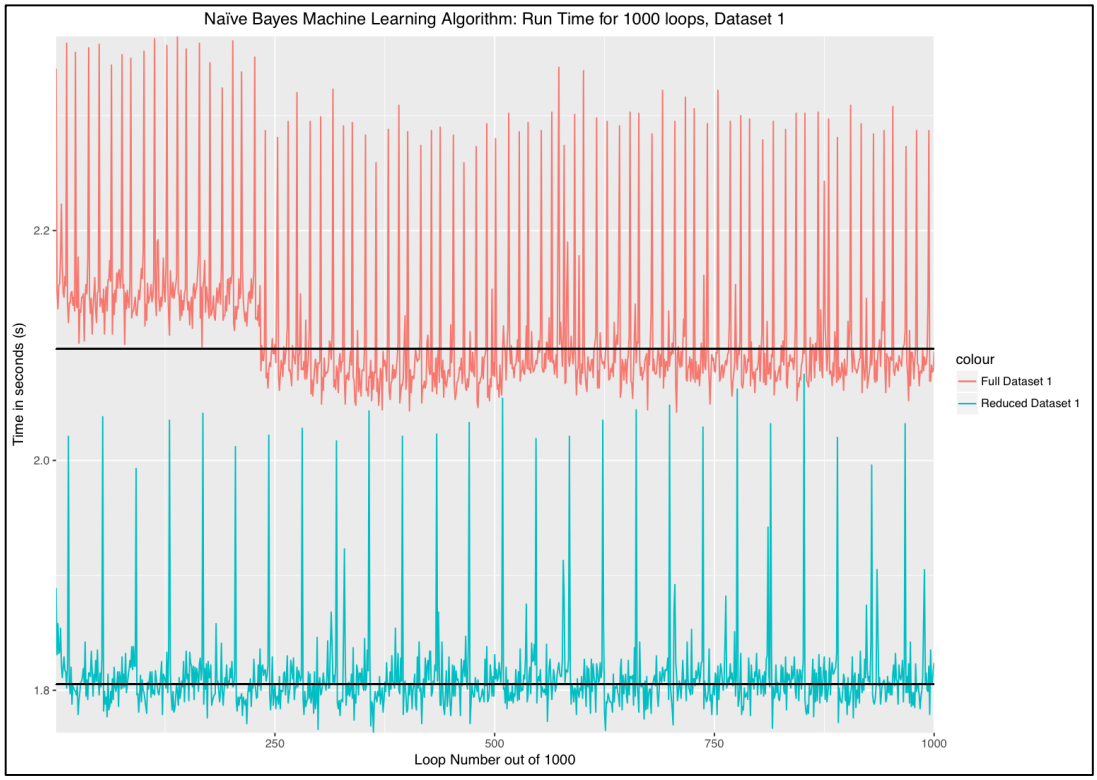


Figure 5.0.2, Illustrates the processing time in seconds(s) for the Naïve Bayes algorithm to process RD1 and FD1. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

5.1.1.2 Naïve Bayes algorithm applied to Datasets Two

The results from applying the Naïve Bayes probability classification algorithm to RD2 and FD2 are presented in this section. Both datasets have a total of 719 sessions of adversary interactions, with the longest sequence of adversary commands recorded at 45 distinct commands in length. The Naïve Bayes algorithm was applied to the datasets using a train and test model. Prior to organising the datasets into the train and test sets, the datasets had been randomly shuffled using a seed value of `123`. With 70% (503 sessions) of the datasets used to train the Naïve Bayes classifier. The feature used to train the datasets was the 15th adversary command in the sequence. The 15th command in the sequence was selected as it is the average command length observed for both datasets. Once the classifier had been trained using the 70% training set, it was applied to the testing set. The test sets comprised of the remaining 30% (216 sessions) of both complete datasets, to test the probability of the trained classifier to correctly classifying instances from the test sets. In the following section, the results obtained from the experiments conducted to test hypotheses *H1*, *H3*, *H5* are presented below.

Classification algorithms use patterns within a given dataset to classify instances, inferring any classified instances are a result of identified patterns within the given dataset by the algorithm. The trained classifier was applied to the test set of both datasets, to verify whether the instances or commands had been classified correctly by the trained classifier. For the purposes of this study, the TP rate was used to determine if additional class patterns can be extracted from the RD2 compared to FD2 (responding to *H1*). The confusion matrix was also used to calculate the TP rates along with the precision of the trained Naïve Bayes classifier to classify instances in the test sets. The TP rates for the RD2 and FD2 had been calculated at 99.505%. In the pre-processing phase (presented in Section 4.2.2), 93.880% of the datasets contained the predominant duplicated sequences of adversarial commands, providing evidence of a script being implemented to interact with the Cowrie SSH honeypot. Subsequently, there are resemblances between RD2 and FD2. The resemblances are also seen between the precision measurements for both datasets as presented in Table 5.2.

The precision of the classified instances are evaluated using the confusion matrix. Table 5.2 shows the precision, accuracy sensitivity, F1 score and error rate values for the Naïve Bayes algorithm applied to the RD2 and FD2. The values in Table 5.2 are on a scale between 0 and 1, where 1 is the greatest value given. Repeatedly, the precision measurements presented in Table 5.2 are the same for both datasets. The precision value used to measure the correctness of the classified instances are greater than the accuracy value used to measure the number of truly classified instances. The accuracy value as seen in Table 5.2 had been recorded at 0.995 while the precision value for both datasets, had been recorded at 0.998. The F1 Score measures the predictive probability of the classified instances had been recorded at 0.799 for both datasets (responding to *H3*).

Table 5.2, Shows the precision, accuracy, sensitivity, F1 score and error rate of the classified instances using the trained Naïve Bayes algorithm for RD2 and FD2. Calculated using the confusion matrices for each dataset, the measurements are on a scale between 0 and 1, where 1 is the highest.

Naïve Bayes	***FD2***	***RD2***
Precision	0.998	0.998
Accuracy	0.995	0.995
Sensitivity (Recall)	0.667	0.667
F1 Score	0.799	0.799
Error Rate	0.005	0.005

Along with the precision of the classified instances, the efficiency of the Naïve Bayes algorithm was assessed by iterating the train and test procedure applied to the datasets 100 times as shown in Figure 5.3 and 1,000 times as presented in Figure 5.4 (responding to *H5*). The mean for both datasets are represented by the black lines after the outliers had been removed. Examining the processing time in seconds(s) for both datasets in Figure 5.3, the processing times are similar with the minimum processing time for FD2 was 0.323s and the minimum recorded processing time for RD2 was lower at 0.326s. That is a difference of 0.003s between the processing times. The maximum processing time for FD2 is 0.697s as seen in Figure 5.3. Whereas, the maximum processing time for RD2 was recorded at 0.468s, a difference of the 0.229s between the maximum recorded processing time for both datasets. To ascertain the mean processing time for the datasets the outliers had been removed prior to calculating the mean processing time for each dataset. A total of nine outliers had been removed before the mean processing time for FD2 was calculated at 0.357s, as shown in Figure 5.3. However, after removing three outliers from the RD2, a mean processing time for Figure 5.3 was calculated at 0.365s. That is a difference of 0.008s as seen in Figure 5.3 for 100 iterations.

Figure 5.4 illustrates the processing time in seconds(s) of the Naïve Bayes algorithm to process RD2 and FD2, iterated 1,000 times. As seen in Figure 5.4 the processing time for both datasets are similar. The maximum processing time recorded for the FD2 was 0.610s and the maximum processing time for RD2 was greater at 0.616s, as seen in Figure 5.4. A difference of 0.006s. Likewise, the minimum processing time recorded for FD2 was 0.320s and the minimum processing time recorded for RD2 was also greater 0.325s. A difference of 0.005s. To ascertain the mean processing time for both datasets, the outliers had been removed to gain a precise efficiency measurement, as represented by the black lines in Figure 5.4. A total of 23 outliers had been removed from FD2 resulting in a mean processing time of 0.337s ascertained. While 36 outliers had been removed from RD2 and a greater mean processing time of 0.348s was calculated, with a difference of 0.011s between both means. The aggregated results are presented in Section 5.3 along with the corresponding hypotheses tested. In the following section the results from applying the Naïve Bayes algorithm to Reduced Dataset Three (RD3) and the respective Full Dataset Three (FD3).

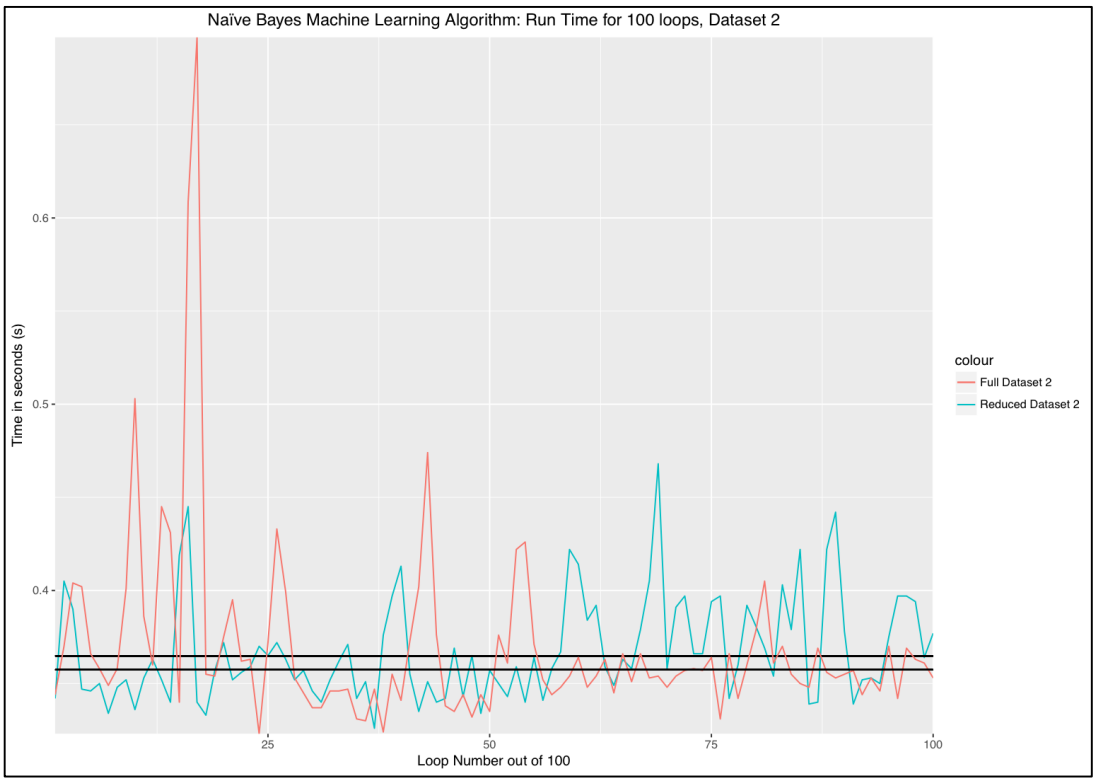


Figure 5.0.3, Illustrates the processing time in seconds(s) for the Naïve Bayes algorithm to process RD2 and FD2. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

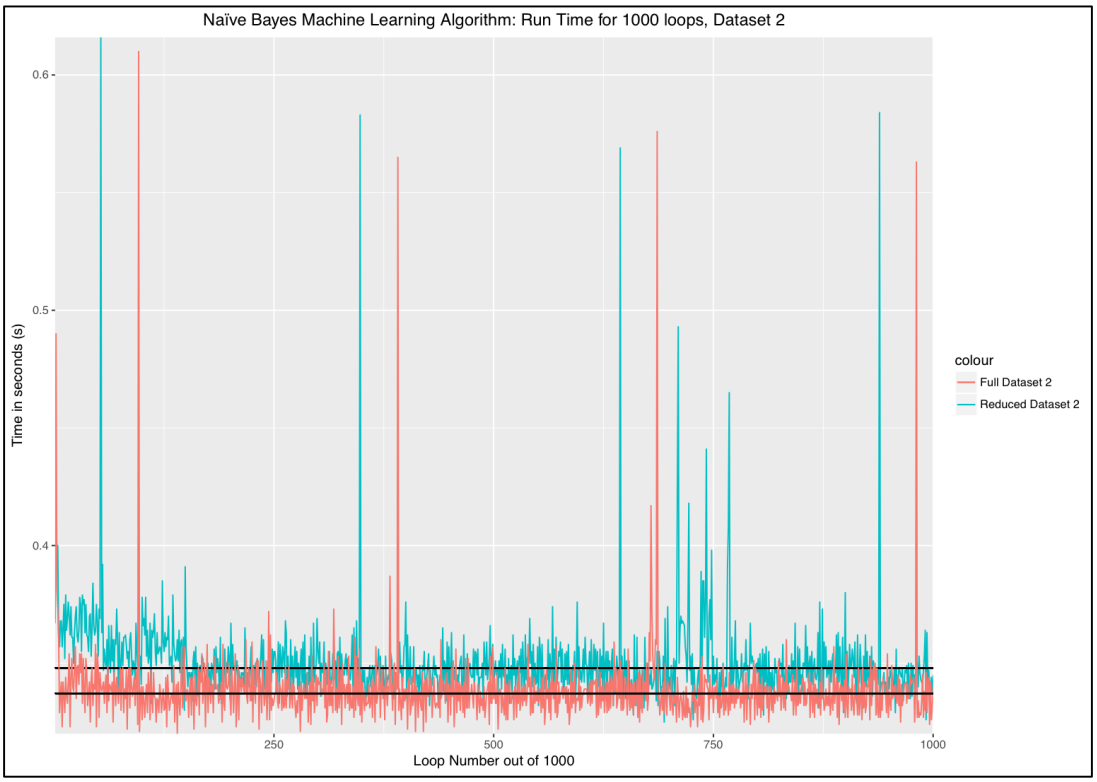


Figure 5.0.4, Illustrates the processing time in seconds(s) for the Naïve Bayes algorithm to process RD2 and FD2. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

5.1.1.3 Naïve Bayes algorithm applied to Datasets Three

In this section, the results from applying the Naïve Bayes algorithm to the RD3 and FD3. Both datasets consisted of 318 sessions of adversary interactions and are randomly shuffled using a seed value of 123, before splitting both datasets into a 70% training set and 30% testing set. The training sets consisted of 223 (70% of the complete datasets) sessions. The feature selected to train the classifier was the 8th adversary command in a sequence, as this was the average length of the adversary command chain for both datasets. Once the classifier had been trained on the training set it was applied to the test set consisting of the remaining 95 (30% of the complete datasets) sessions. The trained classifier would classify the instances in the test set and display the result in a confusion matrix. The confusion matrix was used to assess the identification and precision of the patterns extracted by the Naïve Bayes algorithm from both datasets. The results obtained from the experiments conducted to test hypotheses *H1*, *H3*, *H5* are presented below.

A classification algorithm mines for patterns within a dataset based on the chosen feature. Inferring true patterns are seen if the classification algorithm can predict instances within a given dataset, irrespective of whether the instances are correctly classified. The trained classifier was applied to the test set of both datasets, to verify whether the instances or commands had been classified correctly by the trained classifier. For purpose of this study, the TP rate was used to determine if additional class patterns can be extracted from the RD3 compared to FD3 (responding to *H1*). The TP rate for RD3 and FD3 was recorded at 11.111% for both datasets.

To evaluate the precision of the identified patterns by the Naïve Bayes algorithm, a confusion matrix was used to calculate the measurements as presented in Table 5.3, (responding to *H3*). The values in Table 5.3 are on a scale between 0 and 1, where 1 is the greatest value given. The precision value for RD3 was recorded at 0.161 and was greater than the precision value recorded for FD3 at 0.099, with a difference of 0.008. While the accuracy values of 0.111 recorded and an error rate values of 0.889 recorded for same both datasets. Still, the sensitivity and F1 Score values for RD3 are greater than those recorded for FD3. The variance between the sensitivity values for both datasets was 0.032, and a difference of 0.044 for the F1 score. The variance between the F1 score values is the greater at 0.044, this measurement assesses the predictability of the trained classifier applied to the test dataset.

Table 5.3, Shows the precision, accuracy, sensitivity, F1 score and error rate of the classified instances using the trained Naïve Bayes algorithm for RD3 and FD3. Calculated using the confusion matrices for each dataset, the measurements are on a scale between 0 and 1, where 1 is the highest.

Naïve Bayes	***FD3***	***RD3***
Precision	0.099	0.161
Accuracy	0.111	0.111
Sensitivity (Recall)	0.087	0.119
F1 Score	0.092	0.137
Error Rate	0.889	0.889

The following section assesses the processing time in seconds(s) for the Naïve Bayes algorithm to process RD3 and FD3 (responding to *H5*). Figure 5.5 depicts 100 iterations of the algorithm processing RD3 than FD3, the process encompasses the implemented training and testing model of the classifier. As seen in Figure 5.5, the Naïve Bayes algorithm is efficient at processing the RD3 compared to processing time for FD3. The greatest recorded processing time recorded was for FD3 at 0.981s. The processing time for FD3 ranged from 0.981s to 0.682s while the range for RD3 was between 0.717s and 0.645s, showing FD3 has a greater distribution in the recorded processing time. The black line represents the mean processing time for both datasets after the outliers had been removed. The mean processing time ascertained for FD3 after removing six outliers was recorded at 0.719s. RD3 has a mean processing time of 0.670s once five outliers had been removed, a difference of 0.049s between the means for the datasets. The observations made from Figure 5.5, suggest the Naïve Bayes algorithm is 0.049s efficient at processing the RD3 compared to FD3.

Figure 5.6 depicts the processing time in seconds (s) for 1,000 iterations of the Naïve Bayes algorithm applied to the RD3 and FD3. As seen in Figure 5.5 and Figure 5.6 the algorithm is efficient at processing RD3 than FD3. The greater recorded processing time as shown in Figure 5.6, was for FD3 at 0.937s. The distribution of the processing time for FD3 for 1,000 iterations as recorded between 0.937s to 0.677s, with a mean processing time of 0.707s recorded after 27 outliers had been removed. The distribution of RD3 in Figure 5.6 was greater than that observed in Figure 5.5, ranging between 0.907s and 0.641s. A total of 17 outliers had been removed before a mean processing time of 0.673s was ascertained. The mean for each dataset is represented by the black line shown in Figure 5.6. There is a difference of 0.034s between the means for both datasets. The aggregated results are presented in Section 5.3 along with the corresponding hypotheses tested. In the next section, the results obtained from applying the Markov chain model to the three full and their associated reduced datasets are presented.

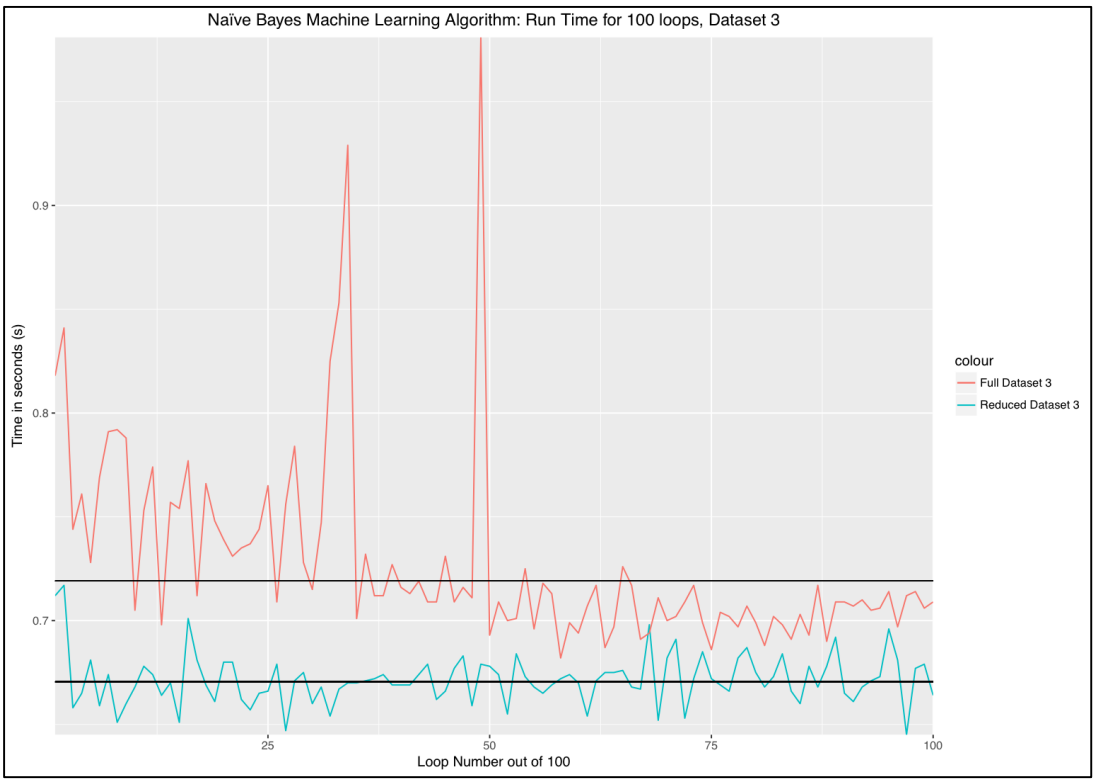


Figure 5.0.5, Illustrates the processing time in seconds(s) for the Naïve Bayes algorithm to process RD3 and FD3. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

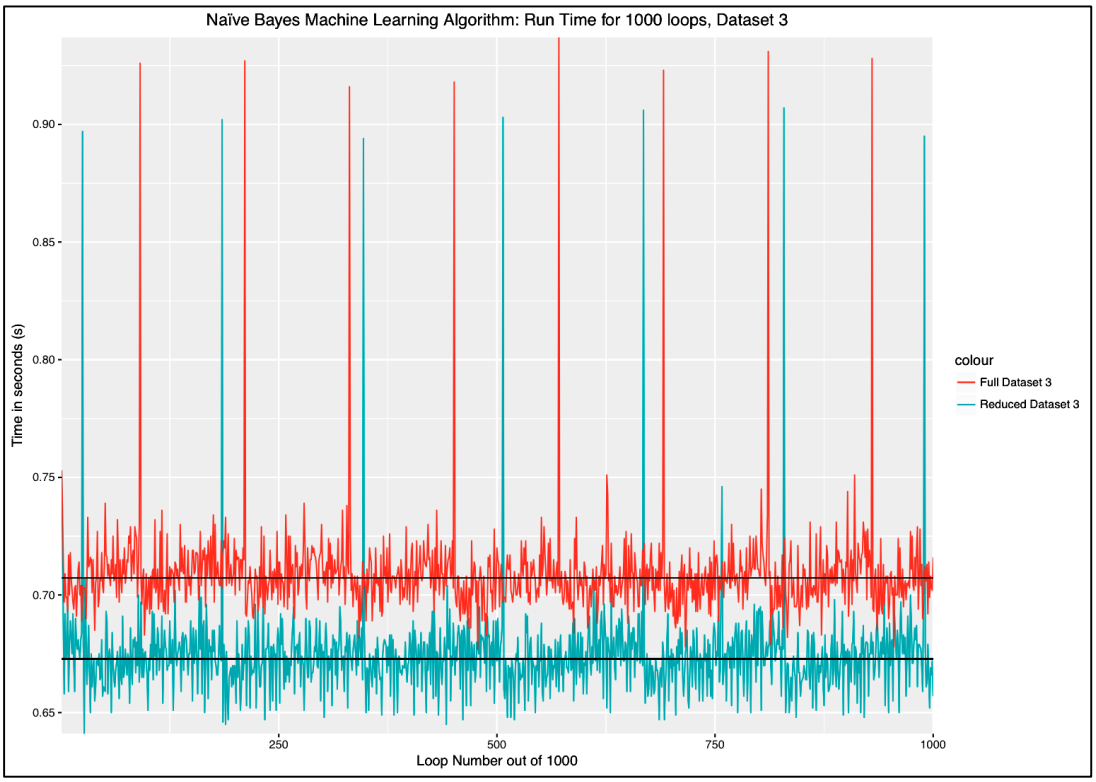


Figure 5.0.6, Illustrates the processing time in seconds(s) for the Naïve Bayes algorithm to process RD3 and FD3. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

5.1.2 Markov Chain Algorithm

A Markov chain algorithm determines the probability of a state transition based on the Markov Property of the given dataset. The Markov Property is based on a stochastic process, were the conditional probability of a state transition is based on the past and current states, not the sequences states observed. As the datasets utilised for this study are of sessions of adversary interactions, the states are the adversary commands. The state transition is the probability of the change in the state from the current command to the next command. The state transition is presented in a square matrix format called the transition matrix. The experiments in the following sections had been developed to test the hypotheses, *H1*, *H3* and *H5* as outlined in Section 3.2

The *R* package markovchain implementation of the Markov chain algorithm was applied to the datasets to allow a true transition matrix to be generated. Prior to applying the Naïve Bayes algorithm to the datasets, the data type was verified to ensure the data was presented as nominal data type (a factor data type in *R*). As well as ensuring any empty cells are represented using *NA* (not applicable) to indicate there is no data within the cell to be processed, whereas an empty cell indicates data is missing. Concluding the pre-processing phase elaborated in Section 4.2.

The Markov chain algorithm uses patterns within the dataset to generate the transition matrix. In order to, assess whether additional patterns can be extracted from the reduced datasets compared to those extracted from their respective full datasets, the degree of freedom was calculated for each dataset. The degree of freedom is the number of independent values observed within a dataset before a subsequent value can be predicted (responding *H1*).

The precision of the transition matrix was used to evaluate the patterns extracted by the Markov chain algorithm. The mean and standard deviation of the standard error matrix, lower endpoint matrix (lower limit) and upper endpoint matrix (upper limit) was used to assess the transition matrix generated by the Markov chain algorithm. The standard error matrix presents the error rate within the transition matrix. The lower endpoint matrix along with the upper endpoint matrix are the lower limit and upper limit of the confidence interval for the transition matrix set at 95%. The difference between the upper limit and lower limit signifies the range where 95% of the calculated probability is true and should not be rejected (responding to *H3*).

The processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for the full and their associated reduced datasets had been iterated 100 and 1,000 times, to ascertain a precise mean efficiency measurement (responding to *H5*). The process was iterated 100 times as opposed to 1,000 times, as evaluating the precision for the algorithm for 1,000 iterations would be

computationally intensive. The convergence of a transition matrix occurs when the probability of a state transition is at an equilibrium and the stated probabilities do not change. Presented below are the results obtained from applying the Markov chain algorithm to the reduced datasets and their respective full datasets.

5.1.2.1 Markov Chain algorithm applied to Datasets One

In this section, the Markov chain algorithm was applied to RD1 and FD1. The Markov chain algorithm was applied to both of the complete datasets to allow the generated transition matrix to truly reflect their relevant datasets. The results obtained from the experiments conducted to test hypotheses *H1*, *H3*, *H5* are presented below.

For the Markov chain algorithm to determine the probability of the next state or command, a transition matrix needs to be formulated. The matrix presents the current state with the probability of the next state occurring. A transition matrix uses patterns identified within a given dataset to determine the probability of the next command occurring. To evaluate if additional patterns can be extracted by the Markov chain algorithm the degree of freedom for RD1 and FD1 are compared (responding to *H1*). The degree of freedom is the number of independent values within a dataset before a subsequent value can be predicted. The lower the degree of freedom, the lower the number of independent values required for a subsequent value to be predicted. The degree of freedom for FD1 as 1.419e+14, while the degree of freedom for RD1 was 1.279e+14. There is a difference of 1.394e+13 between the degree of freedom values for the RD1 and FD1. Signifying less independent values in a sequence are required for a subsequent value to be predicted, suggesting patterns can be observed between fewer values.

The precision of the transition matrix is presented in Table 5.4. Table 5.4, shows the mean and standard deviations of the standard error rate, lower endpoint matrix and upper endpoint matrix (responding to *H3*). The standard error matrix shows the error in the state change for the transition matrix. The mean value indicates the average value of a set of variables, while the standard deviation signifies the spread or divergence of the values from the mean, a low standard deviation denotes the data points are closer to the mean and the dataset has a low probability distribution. The mean standard error rate was greater for FD1 compared to RD1, with a variance of 161.022. The difference in the mean values for the datasets are attributed to a 2,960 square transition matrix generated for FD1 compared to a 405 square transition matrix generated for RD1. The square transition matrix value is determined by the number of possible commands that could occur or the number of unique commands within the dataset. The standard deviations of the standard error rate show the distribution of the error rate from the mean. As seen in Table 5.4, FD1 has a greater standard deviation error rate at 2.026 compared to RD1 that had a recording of 1.257. A difference 0.768 indicates RD1 had a lower spread of data compared to FD1.

Table 5.4, Shows the mean and standard deviation values for the standard error matrix, lower endpoint matrix and upper endpoint matrix of the transition matrices generated by the Markov chain algorithm for RD1 and FD1.

Markov chain		***FD1***	***RD1***
Standard Error matrix	*Mean*	181.758	20.736
	Standard Deviation	2.026	1.257
Lower endpoint matrix	*Mean*	181.072	20.130
	Standard Deviation	0.338	0.415
Upper endpoint matrix	*Mean*	182.060	21.448
	Standard Deviation	3.014	2.455

The lower endpoint matrix is the lower limit of the confidence interval (set at 0.95) for the transition matrices. The mean values of the lower endpoint matrices are also impacted by the square transition matrices generated for both datasets, with a difference of 160.942 between the mean values. While the standard deviation values for the lower endpoint matrix for FD1 was 0.338 and 0.419 for RD1, a difference of 0.082. Indicating the lower endpoint matrix for RD1 is distributed further compared to FD1. The upper endpoint matrix shows the upper limit of the confidence interval (set at 0.95) for the transition matrices. The mean for FD1 is greater compared to RD1, with a difference of 160.613. However, unlike the standard deviation for the lower limit, the upper limit standard deviation value is lower for RD1 as seen in Table 5.4, with a difference of 0.558. Indicating the spread of the upper limit of the confidence value is closer to the mean for RD1. The difference between the upper limit and lower limit signifies the range where 95% of the calculated probability is true and should not be rejected. The difference between the means for the upper limit and lower limit for FD1 was calculated at 0.989. While a greater variance between the mean values for RD1 was calculated at 1.318. Suggesting RD1 has a greater range where 95% of calculated probabilities are true. The observations made from Table 5.4, suggest the transition matrix for RD1 is precise compared to that of FD1. The following figure depicts the processing time for the Markov chain algorithm to generate a converged transition matrix process iterated 100 and 1,000 times.

Figure 5.7 depicts the processing time in seconds(s) for the Markov chain algorithm to generate the converged transition matrices for the RD1 and FD1. The process was iterated 100 and 1,000 times to ascertain a precise efficiency measurement (responding to *H5*). Figure 5.7 and Figure 5.8 are scaled to $\log_{10}$ as the difference between the processing time for RD1 and FD1 is significant. The processing time for RD1 on a non-scaled graph (seen in Appendix B and Appendix C respectively), is not depicted due to the significant difference between the processing times for both datasets. The black lines represent the mean processing time after the outlines had been removed. As seen in Figure 5.7, FD1 has a greater processing time compared to RD1. The greatest recorded processing time for FD1 was recorded at 144.358s compared to the greatest recorded processing time of 3.606s for RD1. While the lowest processing time recorded for RD1 was recorded at 3.288s and the lowest recorded processing was

recorded at 153.005s for FD1. It is also reflected in the mean values for the datasets. A total of 20 outliers had been removed before a mean processing time of 147.179s was calculated for FD1. Whereas, four outliers had been removed from RD1 before a mean processing time of 3.366s was calculated. The finding suggests the Markov chain algorithm efficient at processing RD1 compared to FD1.

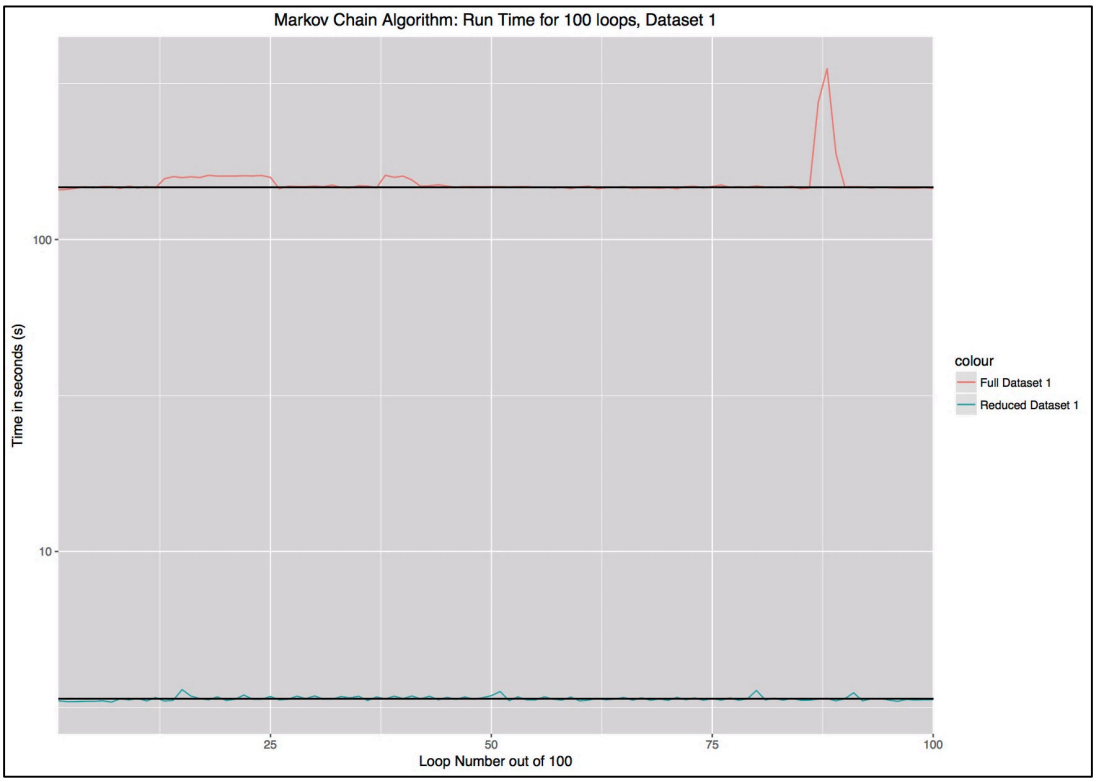


Figure 5.0.7, Illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for the RD1 and FD1, scaled to log_{10}. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

Figure 5.8 depicts 1,000 iterations of the Markov chain algorithm converging the transition matrices for RD1 and FD1. The processing time for FD1 ranges between the greatest times of 148.368s and the lowest processing time of 152.606s. While the processing time for RD1 ranges between the greatest time of 3.605s and the lowest processing time of 2.881s. The black line represents the mean processing time after the outliers had been removed. The mean processing time for FD1 was 147.002s once 186 outliers had been removed and the mean processing time ascertained for RD1 was 3.370s after removing 48 outliers. The observations from Figure 5.7 are the same as the observations seen in Figure 5.8, suggesting the Markov chain algorithm is efficient at processing RD1 compared to FD1. The aggregated results are presented in Section 5.3 along with the corresponding hypotheses tested. In the following section the results from applying the Markov chain algorithm to RD2 and FD2.

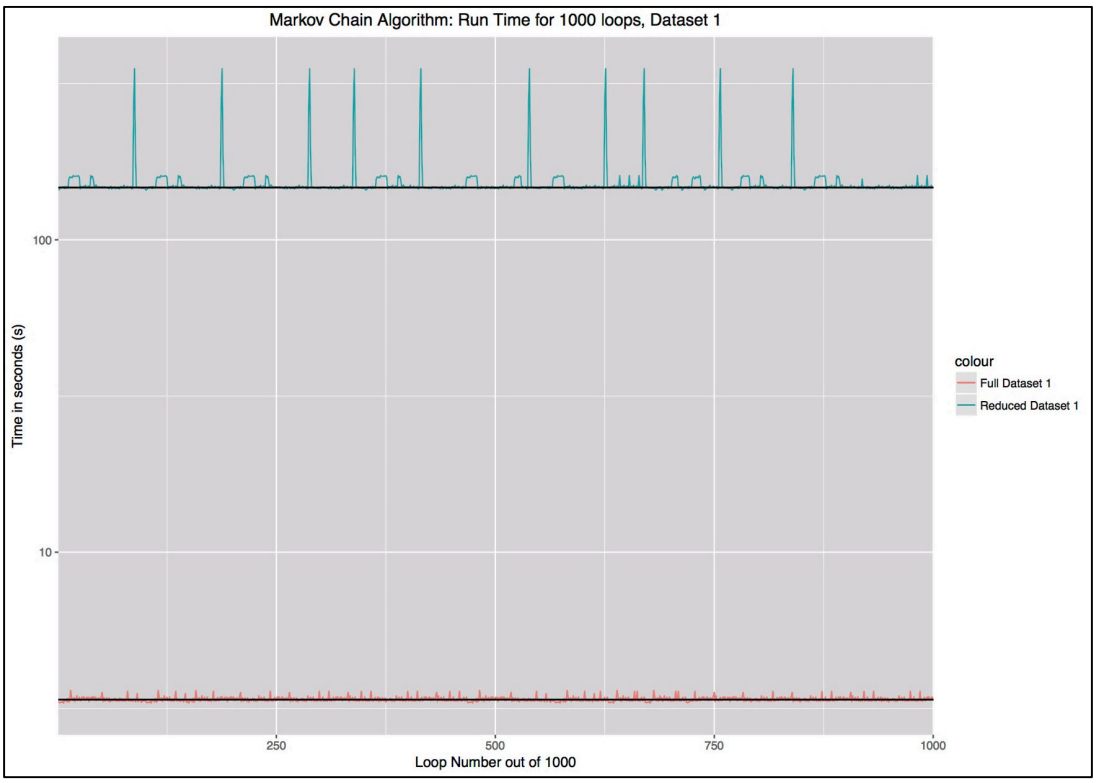


Figure 5.0.8, Illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for RD1 and FD1, scaled to log_{10}. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

5.1.2.2 Markov Chain algorithm applied to Datasets Two

Within this section, the results from applying the Markov chain algorithm to RD2 and FD2. The assessments are based on the transition matrix generated by the Markov chain algorithm. Therefore, both of the complete datasets had been used to generate the transition matrices to truly reflect their relevant datasets. The results obtained from the experiments conducted to test hypotheses *H1*, *H3*, *H5* are presented below.

The degree of freedom had been used to evaluate whether additional patterns can be extracted from RD2 compared to those extracted from FD2 (responding to *H1*). As a transition matrix uses patterns identified within a given dataset to determine the probability of the next command occurring. The degree of freedom is the number of independent values within a dataset before a subsequent value can be predicted. The degree of freedom for FD2 was 778,688,000 and the degree of freedom for RD2 was 667,627,624. With a variance of 111,060,376, even though in the pre-processing phase in Section 4.2.2, it was established datasets two had evidence of a script being utilised to interact with the honeypot. The lower the degree of freedom for RD2 suggests that a pattern has been determined between few commands than FD2. Signifying less independent values are required for a subsequent value to be predicted, suggesting patterns can be observed between fewer commands.

Table 5.5 shows the measurements used to evaluate the precision of the transition matrix for the datasets (responding to *H3*). The standard error matrix shows the error rate for the state transition from the current to the next state or command. The mean standard error rate for FD2 was greater than that of RD2 as seen in Table 5.5, with a difference of 0.224. The difference between the means for both datasets can be attributed to the dimensions of the transition matrices generated. A square transition matrix of 81 was generated for FD2 while a square transition matrix of 45 was generated for RD2. The dimensions of the square transition matrices are determined by the number of possible commands that could occur, this is the same as the number of unique commands within the datasets. The standard deviation value is the spread of the data from the mean, a low standard deviation denotes the data points are closer to the mean and the dataset has a low probability distribution. The standard deviation for RD2 was lower at 0.548 compared to 0.726 for FD2, a difference of 0.179.

Table 5.5, Shows the mean and standard deviation values for the standard error matrix, lower endpoint matrix and upper endpoint matrix of the transition matrices generated by the Markov chain algorithm for RD2 and FD2.

Markov chain		***FD2***	***RD2***
Standard Error matrix	*Mean*	6.653	6.429
	Standard Deviation	0.726	0.548
Lower endpoint matrix	*Mean*	6.156	6.355
	Standard Deviation	0.351	0.421
Upper endpoint matrix	*Mean*	7.123	7.045
	Standard Deviation	0.840	0.714

The confidence interval of 0.95 was set for the transition matrix where the lower limits were shown in the lower endpoint matrix and the upper limits were shown in the upper endpoint matrix. FD2 has a lower mean for the lower endpoint matrix compared to the RD2 as seen Table 5.5, with a difference of 0.199. Additionally, the standard deviation value for the lower endpoint matrix for FD2 at 0.352 is lower than the 0.421 value for RD2, a difference of 0.069. The observation was also seen for the application of the Markov chain algorithm on the datasets one in Section 5.1.2.1. The mean value for the upper limit of the confidence interval for the transition matrix was greater for the FD2 compared to RD2, as seen Table 5.5, with a difference of 0.078. The standard deviation for FD2 was greater at 0.840 compared to 0.714 for RD2, a difference of 0.126. The variance between the upper limit and lower limit signifies the range where 95% of the calculated probability is true and should not be rejected. The variance between the means for the upper limit and lower limit for FD2 was 0.966, while a greater variance between the means for RD3 was calculated at 0.690. Suggesting the RD1 has a greater range where 95% of calculated probabilities are true.

The time in seconds (s) for the Markov chain algorithm to converge the transition matrix for RD2 and FD2 had been iterated 100 times as illustrated in Figure 5.9 (responding to *H5*). The greatest processing

time was recorded for FD2 at 0.861s and the greatest processing time for RD2 was recorded at 0.713s. While the lowest processing time was recorded for RD2 at 0.409s and the lowest recorded processing time for FD2 was recorded at 0.500s. The black line represents the mean processing time once the outliers have been removed. The process was iterated 100 times to ascertain a precise mean processing time. A total of three outliers had been removed from FD2 and a mean of 0.533s was calculated. Whereas two outliers had been removed from RD2 with a mean processing of 0.422s, a difference of 0.111s. The observation made from Figure 5.9 shows the Markov chain algorithm is efficient at processing RD2 compared to FD2.

Figure 5.10 depicts the processing time in seconds(s) for the Markov chain algorithm to process RD2 and FD2, iterated 1,000 times (responding to *H5*). The greatest recorded processing time for FD2 was 0.865s and the lowest recorded processing time was 0.500s. Whereas, the greatest processing time recorded for RD2 was recorded at 0.714s and 0.409s as the lowest recorded processing time. The black lines represent the mean processing after the outliers have been removed. A total of 36 outliers had been removed and the mean processing time of 0.525s was ascertained for FD2. The mean processing time of 0.409s was ascertained for RD2 and a total of 28 outliers has been removed. The observations seen in Figure 5.10 are also seen in Figure 5.9, where the Markov chain algorithm is efficient at processing RD2 compared to FD2. The aggregated results are presented in Section 5.3 along with the corresponding hypotheses tested. In the following section the results from applying the Markov chain algorithm to RD3 and FD3.

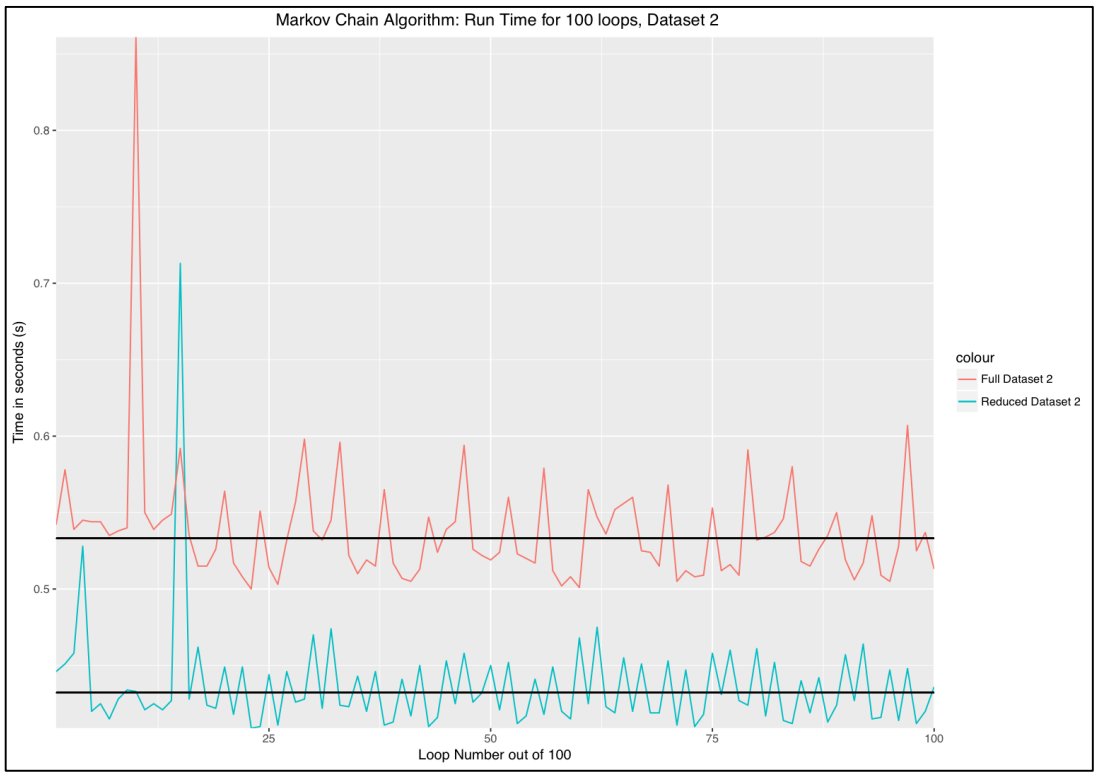


Figure 5.0.9, Illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for RD2 and FD2. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

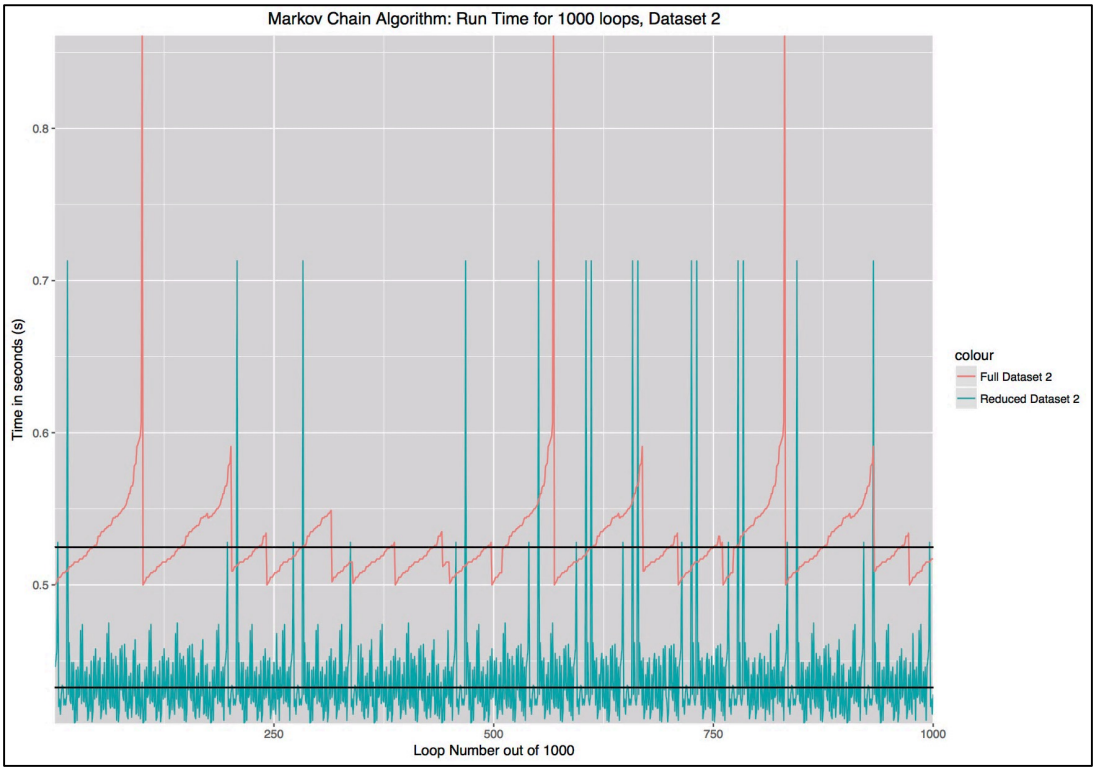


Figure 5.0.10, Illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for the RD2 and FD2. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

5.1.2.3 Markov Chain algorithm applied to Datasets Three

The results of applying the Markov chain algorithm to RD3 and FD3 are presented in this section. The assessments are based on the transition matrix generated by the Markov chain algorithm. Therefore, both of the complete datasets had been used to generate the transition matrices to truly reflect their relevant datasets. The results obtained from the experiments conducted to test hypotheses *H1*, *H3*, *H5* are presented below.

The Markov chain algorithm identifies patterns within the given dataset to calculate the probability of the next state or in the case of this study a command occurring (responding to *H1*). The probabilities of the state transitions are presented in a transition matrix. To investigate whether additional patterns can be extracted from RD3 compared to those extracted from FD3, the degree of freedom was analysed. The degree of freedom is the number of independent variables required before a subsequent variable can be predicted. A lower degree of freedom indicates a lower number of independent variables required before a pattern is formulated. The degree of freedom for FD3 was 1.943e+15 and was greater than the degree of freedom for RD3 at 3.070e+14, this is a difference of 1.636e+15. From the results calculated the degree of freedom, RD3 requires less independent values to formulate a pattern to predict the subsequent values. The precision of the transition matrices calculated using the patterns identified in the datasets by the Markov chain algorithm is presented in Table 5.6.

Table 5.6 shows the precision of the transitions matrices generated for the datasets (responding to *H3*). A standard error matrix encompasses the error rate from the converged transition matrix, showing the probability of the next command occurring based on the current command. As observed in Table 5.6 and Table 5.6 the full dataset has a greater mean value for the standard error matrix, lower endpoint matrix and upper endpoint matrix, this can be attributed to a 748 square transition matrix generated for the dataset. The dimension of the transition matrix is determined by the number of unique commands within the dataset. A 338 square transition matrix was generated from RD3. The standard deviation of the standard error matrix for FD3 was 2.623. The standard deviation is the spread of data within a given dataset, a low standard deviation denotes the data points are closer to the mean and the dataset has a low probability distribution.

Table 5.6, Shows the mean and standard deviation values for the standard error matrix, lower endpoint matrix and upper endpoint matrix of the transition matrices generated by the Markov chain algorithm for RD3 and FD3.

Markov chain		***FD3***	***RD3***
Standard Error matrix	*Mean*	77.831	36.778
	Standard Deviation	2.626	1.863
Lower endpoint matrix	*Mean*	77.009	36.024
	Standard Deviation	0.112	0.110
Upper endpoint matrix	*Mean*	78.108	37.299
	Standard Deviation	3.652	3.010

As seen in Table 5.6, the standard deviation for the standard error matrix is lower for RD3 at 1.863. The lower endpoint matrix and upper endpoint matrix is the lower limit and upper limit of the confidence interval set at 0.95 of the transition matrices generated. The variance between the upper limit and lower limit signifies the range where 95% of the calculated probability is true and should not be rejected. The variance between the means for the upper limit and lower limit for FD3 was 1.099 while a greater variance between the means for RD3 was calculated at 1.275. Suggesting the RD3 has a greater range where 95% of calculated probabilities are true. The standard deviation of the lower endpoint matrix and upper endpoint matrix for RD3 was lower than FD3, as seen in Table 5.6. Indicating the data points are closer to the mean values and the dataset have a low probability distribution. A comparison of the values shows RD3 has a lower mean value for the standard error matrix, lower endpoint matrix and upper endpoint matrix. However, the variance between the standard deviation for the lower endpoint matrix and upper endpoint matrix show RD3 has a greater range where 95% of the calculated probabilities can be considered true. Along with the precision of the transition matrix, the efficiency of the Markov chain algorithm to converge the transition matrices for the datasets are assessed and presented in Figure 5.9 Figure 5.10.

Figure 5.11 shows the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for RD3 and FD3, iterated 100 times (responding to *H5*). The processing time for all 100 iterations for FD3 are greater than RD3, with the greatest processing time at 3.021s recorded for FD3. Whereas, the greatest recorded processing time for RD3 was 0.952s. The lowest recorded processing time was recorded for RD3 at 0.593s and 2.456s for FD3. To ascertain a precise processing time for both datasets the outliers had been removed before the mean processing time was calculated. The mean processing time for both datasets are represented by the black lines shown in Figure 5.11. A total of three outliers had been removed from FD3 before a mean of 2.575s was ascertained. While six outliers had been removed from RD3 and a mean value of 0.616s was calculated. From the ascertained mean processing time, it can be stated that the Markov chain algorithm is 1.959s efficient at converging the transition matrix for RD3 compared to FD3.

The processing time in seconds(s) for the Markov chain algorithm to process the RD3 and FD3 had been iterated 1,000 times as illustrated in Figure 5.12 (responding to *H5*). The greatest processing time recorded for FD3 was 3.021s and 2.451s was recorded as the lowest processing time. Whereas, the greatest recorded processing time for RD3 was 0.956s and 0.598s as the lowest recorded processing time. A total of 237 outliers had been removed from FD3 before a mean of 2.510s was ascertained. The mean processing time for RD3 was recorded at 0.433s after 28 outliers had been removed. The aggregated results are presented in Section 5.3 along with the corresponding hypotheses tested. In the following section, the results from applying the association rule mining algorithms to the reduced datasets and their respective full datasets are presented.

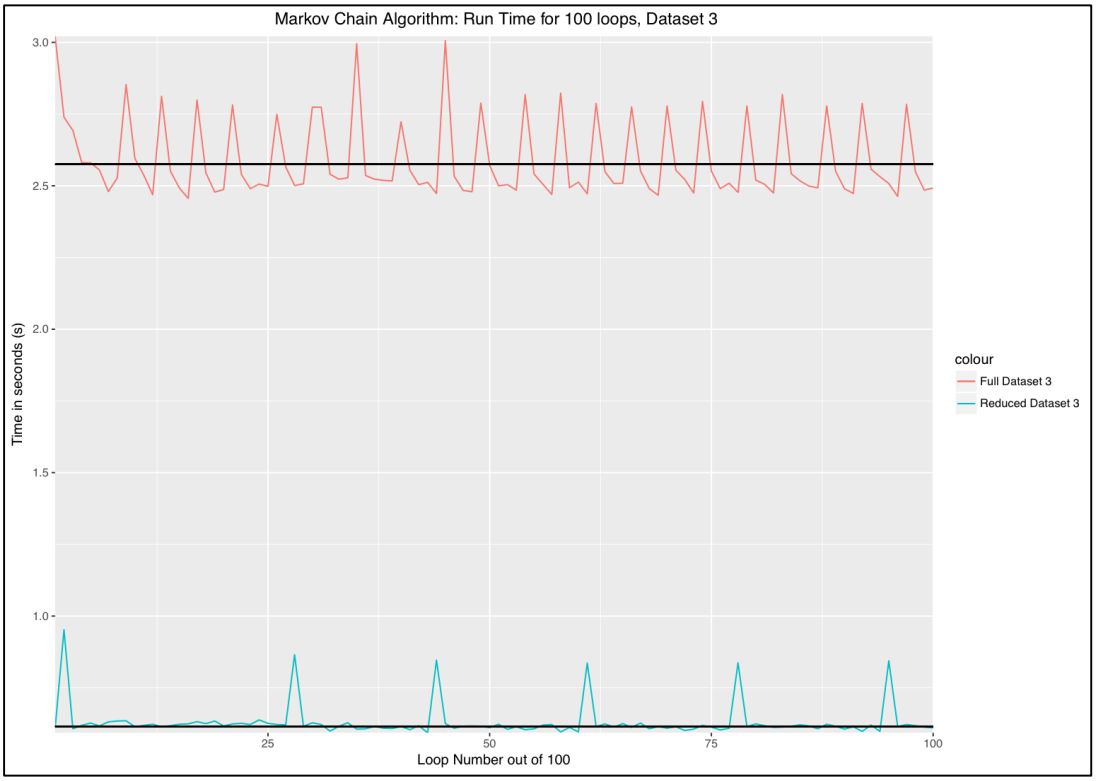


Figure 5.0.11, Illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for RD3 and FD3. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

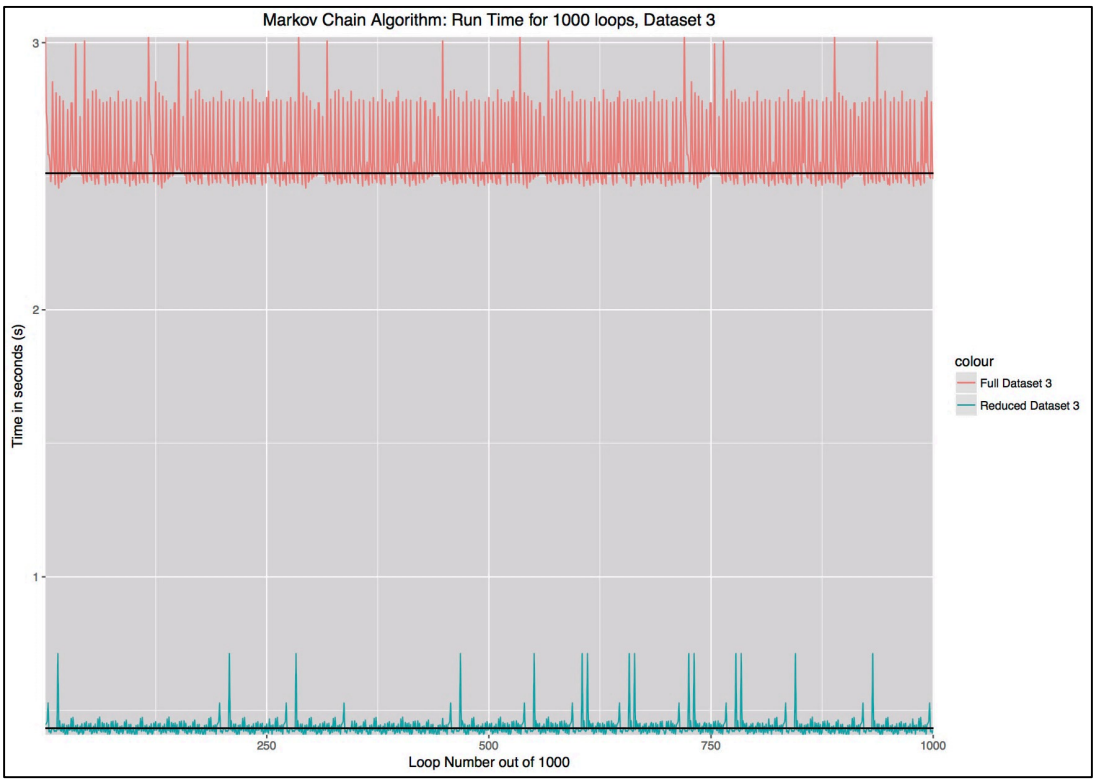


Figure 5.0.12, Illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for RD3 and FD3. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

5.2 Association Rule Mining

Within this section, the results obtained from applying the association rule mining algorithms to the six pre-processed datasets are presented. The selected association rule mining algorithms are Apriori and the Equivalence Class Transformation (Eclat) algorithms, both calculate the probability of a sequence of objects occurring based on the frequency they appear in the given dataset. The extracted sequence of objects and the probability of the occurrence are formulated into rules. Each rule has three values of interest used to determine the probability of the extracted rule occurring, the values are support, confidence and lift.

The support value determines the probability of a rule occurring in the given dataset (Eq. 3.9). A support value between $0 \leq 1$ is given to each rule. Whereby a support value of 1 suggests the rule is relevant in the given dataset, whereas a support value of 0 infers a low probability of the rule occurring in the given dataset.

The confidence of a rule measures the probability of observing an object within a sequence, based on the presences of an initial object/s (Eq. 3.8). For example, if the initial object $\mathcal{X}$ is observed within a dataset, the confidence is the probability of $\mathcal{Y}$ occurring in the sequence if $\mathcal{X}$ is represents. Similar to the support values, the confidence of a rule is measured on a scale between $0 \leq 1$. A confidence value

of 1 denotes y will be observed in a sequence if x is represents and a confidence value of 0 suggests y will not be observed in a sequence with x.

The lift value given to a rule measures the relationship between the objects within a sequence (Eq. 3.10). Traditionally, a lift value between $0 \leq 1$ is given to a rule where, a lift value of 1 suggest objects within the rule are dependent on each other, inferring the sequence of objects observed in the given rule is not random. While a lift value of 0 signifies the objects in the sequence are independent from each other and the rule may not be applicable. However, the *R* implementation of the arules package used for the Apriori and Eclat algorithms has a lift value between $0 \leq \infty$ (infinity) for each rule. A lift leaves near to 1 indicates the x and y objects in a rule are certain to appear together in a sequence.

A transaction form dataset is commonly applied to association rule algorithms, for this study the objects in the pre-processed datasets are commands utilised by adversaries, while the sequences are the chain of commands used by adversaries. The rules extracted by the association rule algorithms are the patterns identified in the chain of adversary's commands. Experiments in this section are related to hypotheses *H2*, *H4* and *H6* (elaborated in Section 3.2.)

Patterns are identified through the extraction of rules by the association rule mining algorithms when applied to the reduced datasets and their respective full datasets. The association rule mining algorithms mine for frequent item sets or instances within a dataset to formulate rules. To ascertain if additional patterns can be extracted by the algorithms from the reduced datasets compared to their respective full datasets, the redundant rules had been removed the number of rules remaining had been compared (responding to *H2*).

The precision of each rule is determined based on the strength of the rules extracted (responding to *H4*). The values of interest are support, confidence and lift. These values are used to determine the precision of the patterns extracted. Once the redundant rules have been removed, the strongest 50 rules based on the support value had been extracted and their precision evaluated, based on the three values of interest. The strongest 50 rules are based on the support value since the support value is used to determine the applicability of the rule to the given dataset.

The efficiency of the association rule mining algorithms to extract the rules from the reduced datasets and their respective full datasets are tested through comparing the time in seconds(s), for the extraction process to occur (responding to *H6*). To ensure the efficiency of the algorithms to process the datasets are representative, the process was iterated 100 and 1,000 times. The process was iterated 100 times as opposed to 1,000 times, as evaluating the precision for the algorithm for 1,000 iterations would be computationally intensive. The mean of the interactions had been taken after removing any outliers.

Thereafter the means had been compared between the reduced datasets and their respective full datasets. In the subsequent sections, the results obtained from the test conducted on utilising the Apriori and Eclat machine learning algorithms to the three reduced datasets and their respective full datasets are presented.

5.2.1 Apriori algorithm

The results obtained from applying the Apriori algorithm to the three reduced datasets and their respective full sequence of commands datasets are presented. There are two main stages to the Apriori implementation in *R*. The first is extracting the rules from the datasets, followed by trimming the rule set by removing the redundant rules from the extracted rule sets. The *R* package `arules` implementation of the Apriori algorithm was used for this study. Prior to executing the Apriori algorithm on the datasets, the format of the dataset was checked to ensure the dataset was in a factor or a nominal format, then shuffled using a random seed of `123`.

As each of the three acquired databases are distinct the default parameters had been adjusted based on the dataset presented. The default parameters for the Apriori implementation in *R* to mine rules are as followed, a minimum support value of 0.1, minimum confidence value of 0.8 and a maximum length of rules with 10 objects. An initial assessment was conducted on a randomly selected 25% of entries from the three full datasets, to determine the parameters of the values of interest. These parameters would also be applied to their associated reduced datasets. The parameters for each value of interest for each dataset is presented in Table 5.7.

As shown in Table 5.7, the minimum length of a rule was to be set at 2 as there ought to be more than one command in an extracted rule. The maximum length of a rule was to be set at 10, which is the default value for the parameter. The initial assessment of FD1 showed the support and confidence was to be set at 0.05 as this was the minimum support and confidence values for the sampled data. The initial assessment for FD2 showed the minimum support and confidence parameters for the extracted rules was to be set at 0.8. While the initial experiment for FD3 showed the support and confidence values was to be set at 0.01. The parameters for the values of interest as shown in Table 5.7, had been applied to the three reduced datasets and their respective full sequence of commands datasets. The results obtained are presented below.

Table 5.7, Shows the results from the initial analysis conducted on the three distinct datasets, in order to determine the minimum support and confidence values to be set along with the length of the rules to be extracted.

Dataset	***Number of samples taken, 25%***	***Minimum Support***	***Minimum Confidence***	***Maximum length***	***Minimum length***
Dataset one (1128)	282	0.05	0.05	10	2
Dataset two (719)	180	0.90	0.90	10	2
Dataset three (318)	80	0.01	0.01	10	2

5.2.1.1 Apriori algorithm applied to Datasets One

In this section, the results of applying the Apriori algorithm to RD1 and FD1 are presented below. Both datasets have a total of 1,128 recorded sessions of adversary interactions with a maximum sequence length of 177 commands. The initial assessment conducted on a random 25% sample of the FD1, showed the minimum support and confidence was to be set at 0.05. Table 5.8, shows the number of extracted rules for dataset one and the number of unique rules within the extracted rule set using the defined parameters in Table 5.8 (responding *H2)*. 1,016 rules had been extracted from FD1 with 98 unique rules, that was 90.354% reduction in the number of rules. Whereas the reduced dataset had a 99.577% reduction in the number of rules after removing the duplicated rules. A total of 447,538 rules had been extracted from the RD1 and 1,893 unique rules remained after removing the duplicate rules.

Table 5.8, Shows the total number of rules extracted, number of unique rules after removing the duplicate rules and the percentage of duplicate rules extracted by the Apriori algorithm applied to RD1 and FD1.

Apriori algorithm	***FD1***	***RD1***
Total number of rules extracted	1,016	447,538
Number of unique rules after removing the duplicated rules	98	1,893
Percent of duplicate rules	90.354%	99.577%

Figure 5.13, is a scatter plot showing the support and corresponding confidence values for each of the unique rules extracted from FD1. The lift is represented by the colour scale of the data points, the darker the colour the greater the lift. From Figure 5.13 of the FD1 the unique rules are in the cluster within the rule predominantly having a low support value, but varying values of confidence. With the confidence value of the rules cluster about confidences value of 1, 0.7 and 0.5. Compared to Figure 5.14 a scatter plot of the unique rules extracted from the RD1. Where the rule rules cluster around a support value of 0.6 and a confidence value between 1 and 0.8. The mean support value for Figure 5.13 is 0.073 and a mean confidence value of 0.842. While the mean support value for Figure 5.14 is lower at 0.071 and a greater mean confidence value of 0.874 compared to the FD1. The mean lift is greater for Figure 5.13 is 10.119 and 9.178 for Figure 5.14.

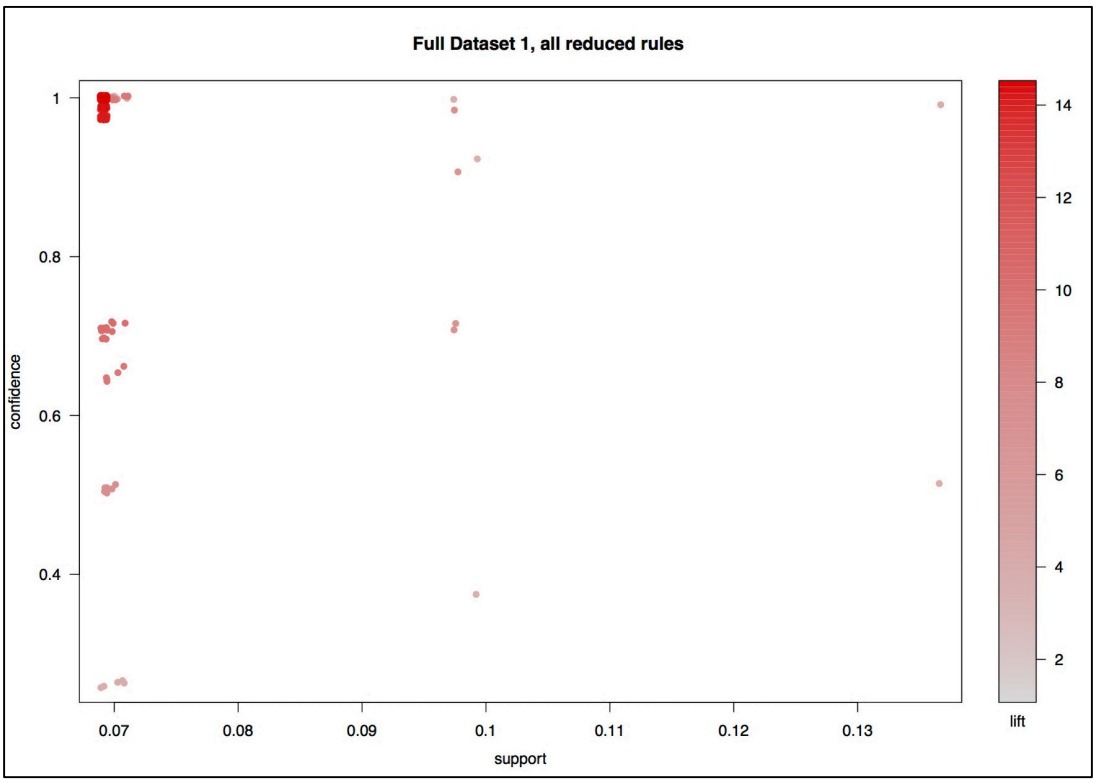


Figure 5.0.13, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Apriori algorithm from FD1.

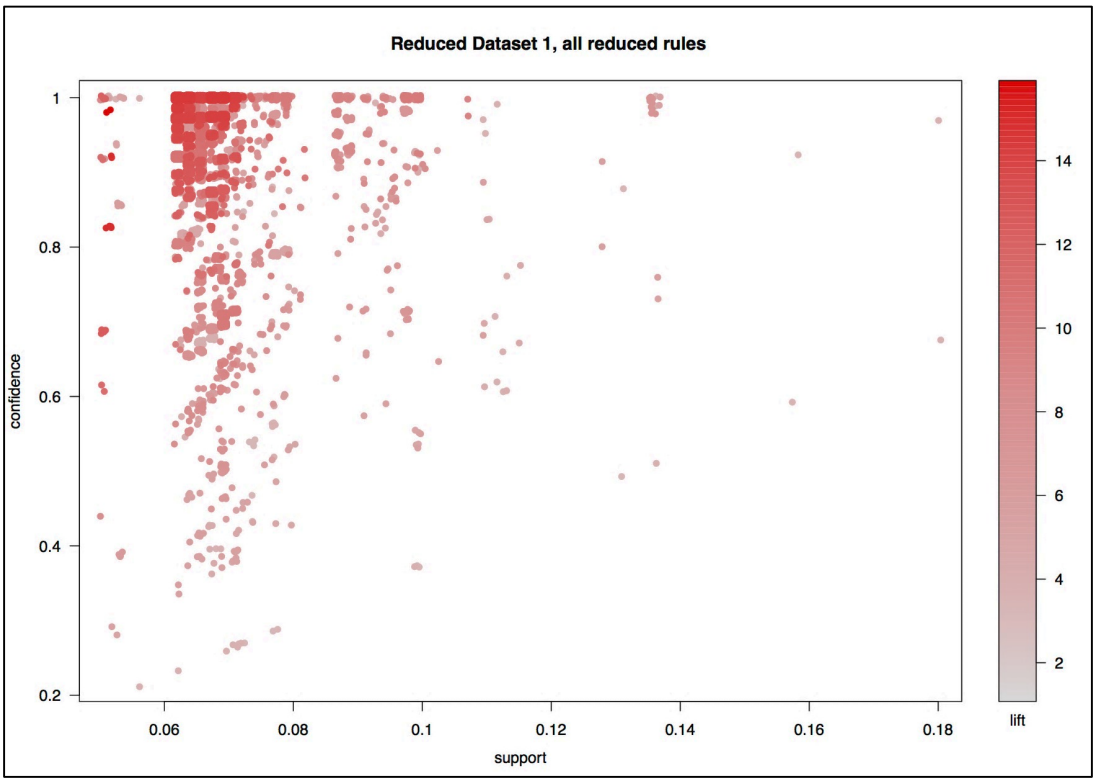


Figure 5.0.14, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Apriori algorithm from RD1.

To analysis the precision of the rules extracted by the Apriori algorithm on RD1 and FD1, the top 50 unique rules based on the support are evaluated. As shown in Figure 5.15, the top 50 unique support based rules extracted from the FD1, range in support between 0.07 and 0.14. However, the is rules have a greater confidence rate with a cluster between 0.9 and 1, but the associated support is 0.07. Figure 5.16 shows the top 50 unique support rules for the RD1, with the support ranging between 0.01 and 0.1 yet the confidence is more scattered compared to Figure 5.15. Table 5.9 shows the support, confidence and lift of the top 50 rules based on the support value. Comparing the distribution of the rules by considering at the minimum, first quartile, medium, mean, third quartile and maximum values for RD1 and FD1 (responding to *H4*).

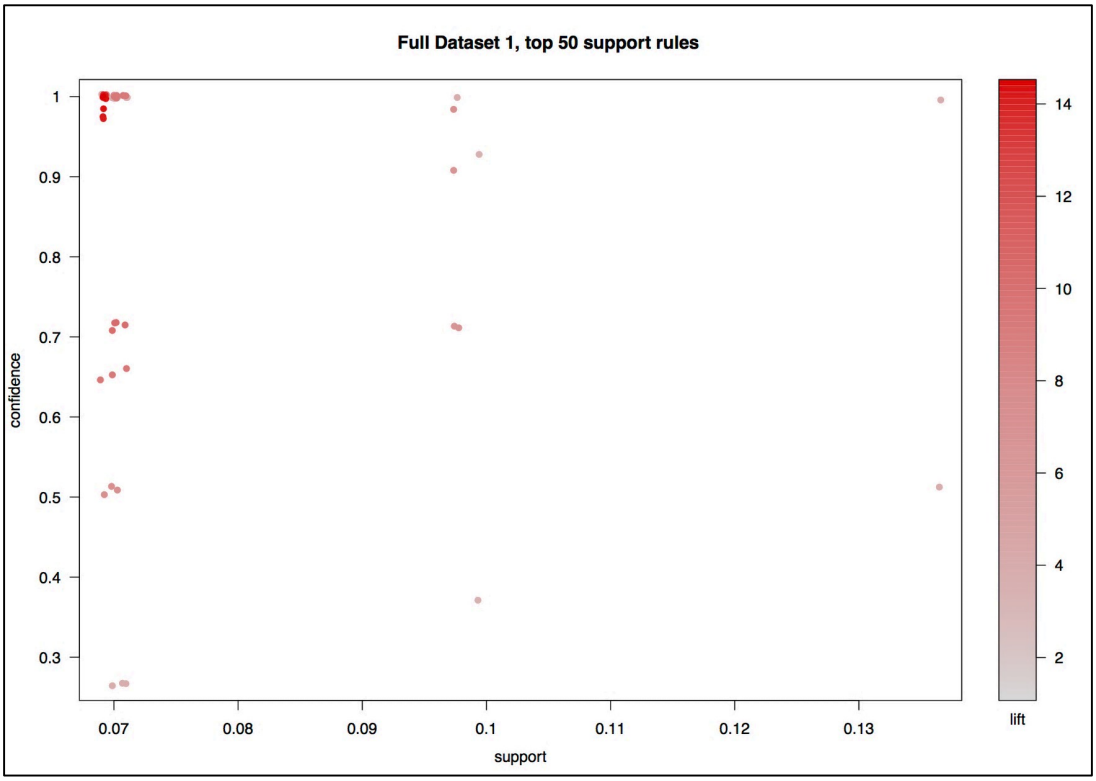


Figure 5.0.15, Illustrates a scatter plot of support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from FD1.

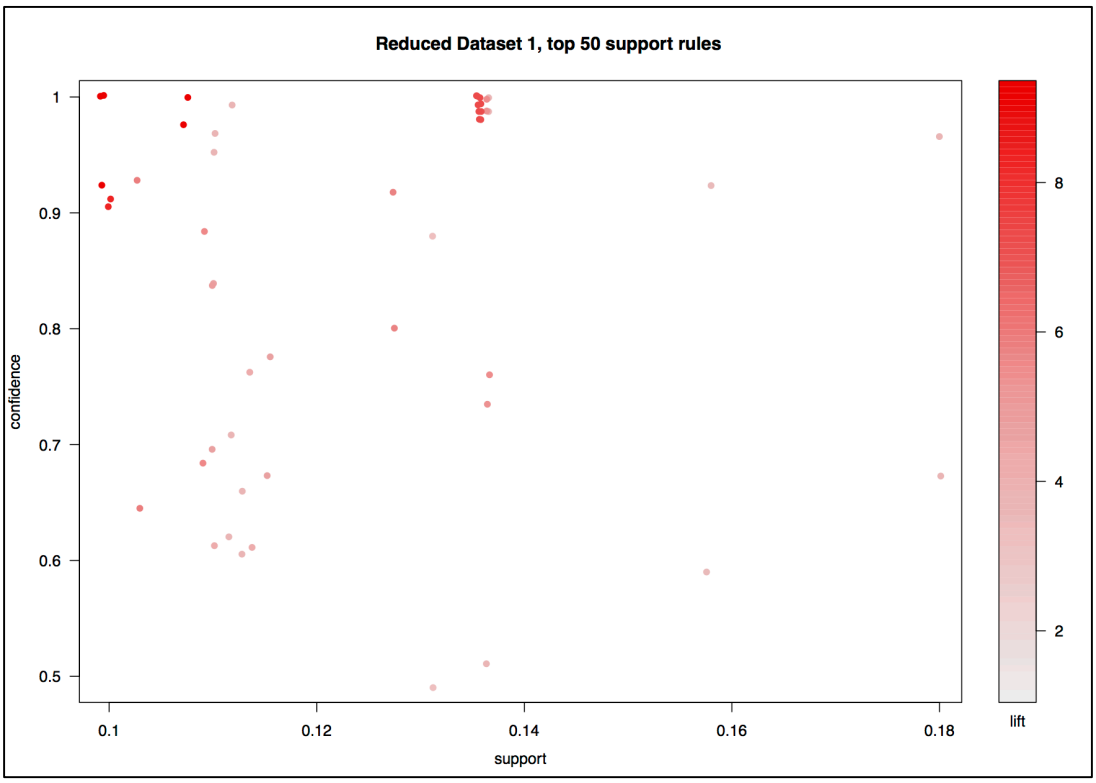


Figure 5.0.16, Illustrates a scatter plot of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from RD1.

As represented in Table 5.9, the RD1 has the greater support measurements compared to the FD1. However, FD1 has a greater first quartile measurement at 0.711 compared to the 1st quartile value of 0.699 for RD1. As well as a greater medium confidences value of 1 compared to 0.914 of the RD1. Additionally, the confidence value of the third quartile of the FD1 has a greater with a value of 1 compared to 0.987 of the RD1. Yet the RD1 has a greater confidence mean value of 0.846 compared to the FD1 confidence mean of 0.844 and a greater minimum of 0.492 compared 0.263 of the FD1. When considering the lower the lift, the objects or commands with an extracted rule are dependent on one another. The RD1 has lower lift values for all measurements, inferring the commands in the rules are likely to occur together when compared to the full dataset.

Table 5.9, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Apriori algorithm applied to RD1 and FD1.

Apriori algorithm applied to the first datasets	***Support***		***Confidence***		***Lift***	
	Full	**Reduced**	**Full**	**Reduced**	**Full**	**Reduced**
Minimum	0.069	0.099	0.263	0.492	3.469	3.301
First quartile	0.069	0.110	0.711	0.699	3.748	3.704
Medium	0.070	0.115	1.000	0.914	7.277	5.303
Mean	0.077	0.124	0.844	0.846	7.848	5.515
Third quartile	0.071	0.136	1.000	0.987	9.322	7.184
Maximum	0.137	0.180	1.000	1.000	14.462	9.322

The top 50 unique support rules extracted from RD1 and FD1, are visually illustrated in Figure 5.17 and Figure 5.18 respectively. The confidence levels of the rules are represented by the arrows connecting the commands and nodes. The connections represent the sequence of commands within the extracted rules. The size of the nodes represents the support for the rule, the larger the size the greater the support for that particular rule. While the colour of the nodes represents the lift of the rule, the darker the colour the greater the lift. From the graphical representation of the top 50 support rules, in Figure 5.17 for the FD1 shows the rules are interconnected with four key commands in eight different sequence placements forming the rules. However, the graphical representation of the top 50 support rules in Figure 5.18, shows there are four clusters of commands sequences for the RD1. There are ten key commands in 15 different sequence placements producing the top 50 rules.

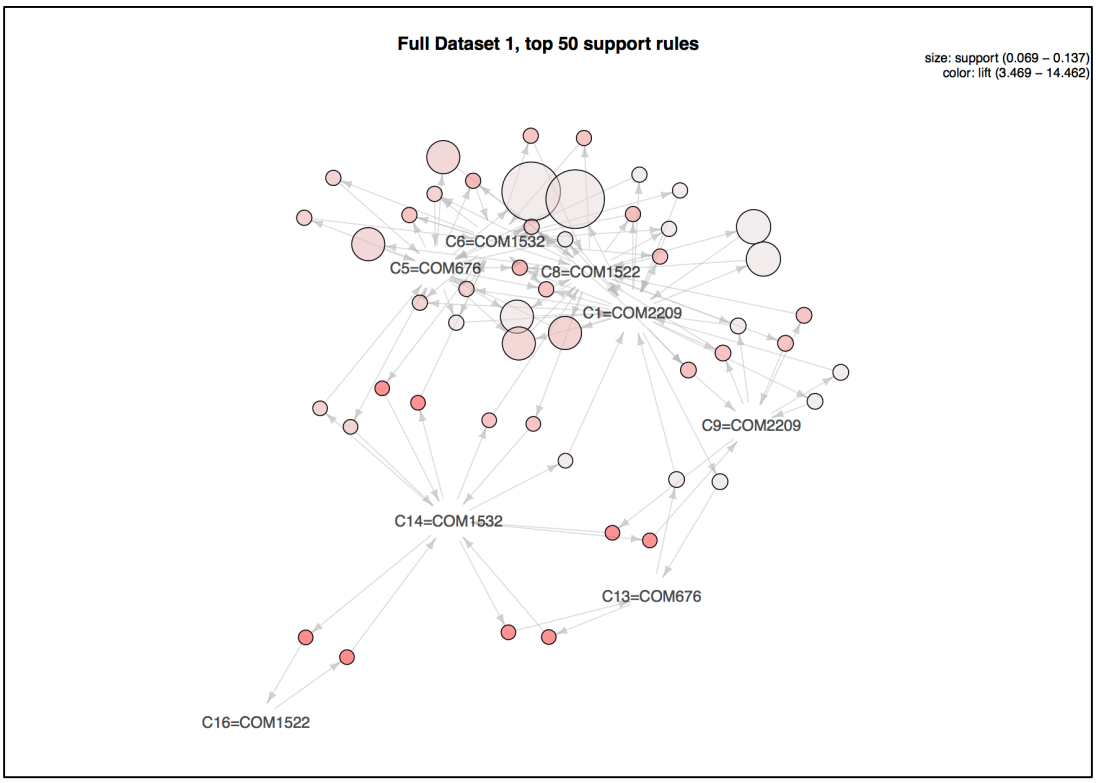


Figure 5.0.17, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from FD1.

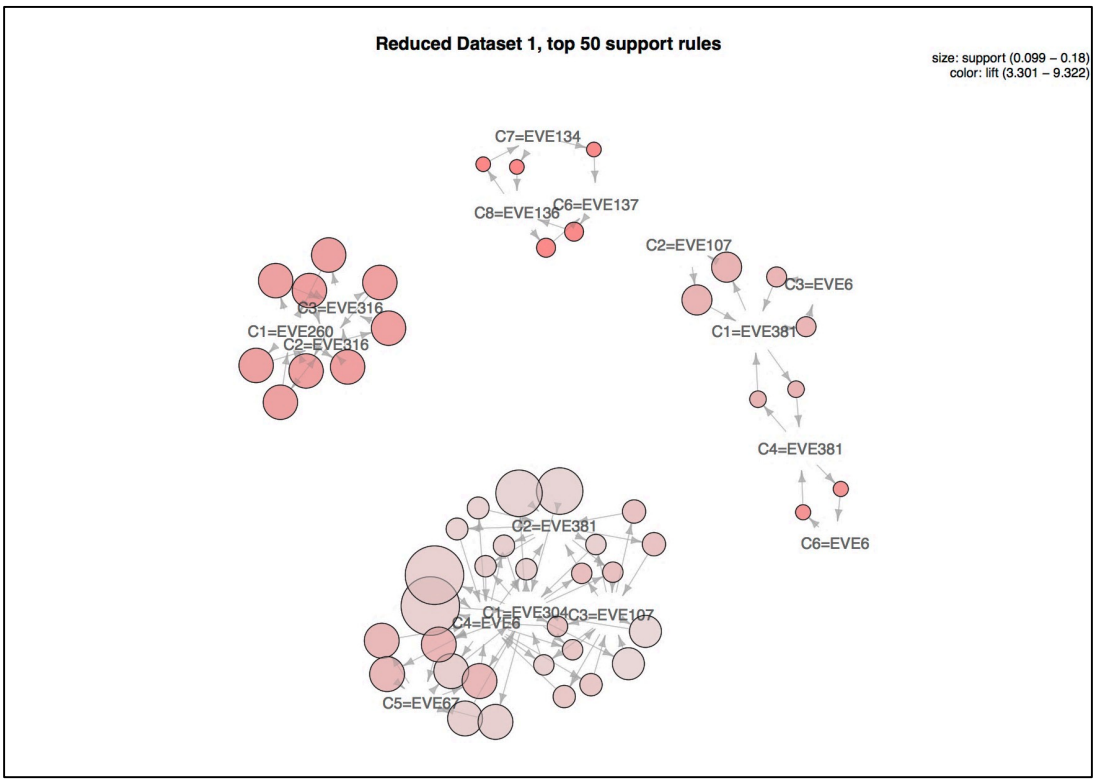


Figure 5.0.18, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from RD1.

To verify a representative efficiency measurement of the Apriori algorithm to process RD1 and FD1, the process was iterated 100 and 1,000 times (responding to *H6*). Figure 5.19 and Figure 5.20 show the processing time in seconds for each loop number measure for 100 and 1,000 iterations respectively. The black line shows the mean processing time for RD1 and FD1, after removing the outliers.

All recorded data points for the FD1 in Figure 5.19 to process the dataset have a greater processing time compared to those recorded for the RD1. The greatest recorded process time for FD1 was 0.210s and 0.079s for the RD1. While 0.174s was recorded as the lowest processing time for full datasets one and 0.051s for RD1. For Figure 5.19 a total of six outliers had been removed from FD1 with a mean of 0.183s. Two outlying data points had been removed in Figure 5.19 for the RD1 with a recorded mean of 0.058s. The data points illustrated in Figure 5.19 RD1 is efficient then FD1 when applied to the Apriori association rule mining algorithm.

The observations made from Figure 5.19 are applicable to Figure 5.20 illustrating the processing time of 1,000 data points. All data points recorded for FD1 in Figure 5.20 had a greater processing time recorded compared to the recorded processing time of RD1. Particularly four data points spread evenly had recorder greater than expected processing time for FD1. With the greatest recorded processing time at 0.431s with the RD1 recording 0.094s. The lowest recorded processing for FD1 in illustrated in

Figure 5.20 was 0.173s, while the reduced dataset recorded 0.048s. 51 outliers from the 1,000 iterations for FD1 had been removed resulting in a mean of 0.184s compared to 0.058s recorded for RD1 with 46 outliers removed. The aggregated results are presented in Section 5.3 along with the corresponding hypotheses tested. In the following section the results from applying the Apriori algorithm to RD2 and FD2.

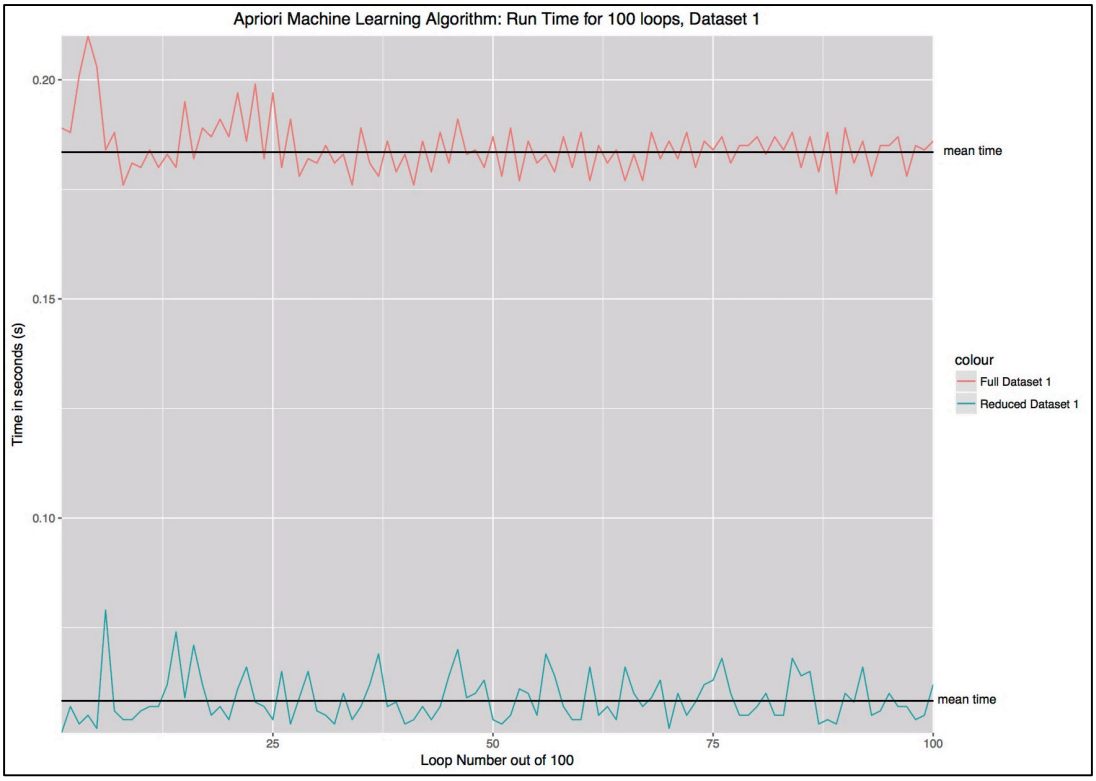


Figure 5.0.19, Illustrates the processing time in seconds(s) of the Apriori algorithm process RD1 and FD1. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

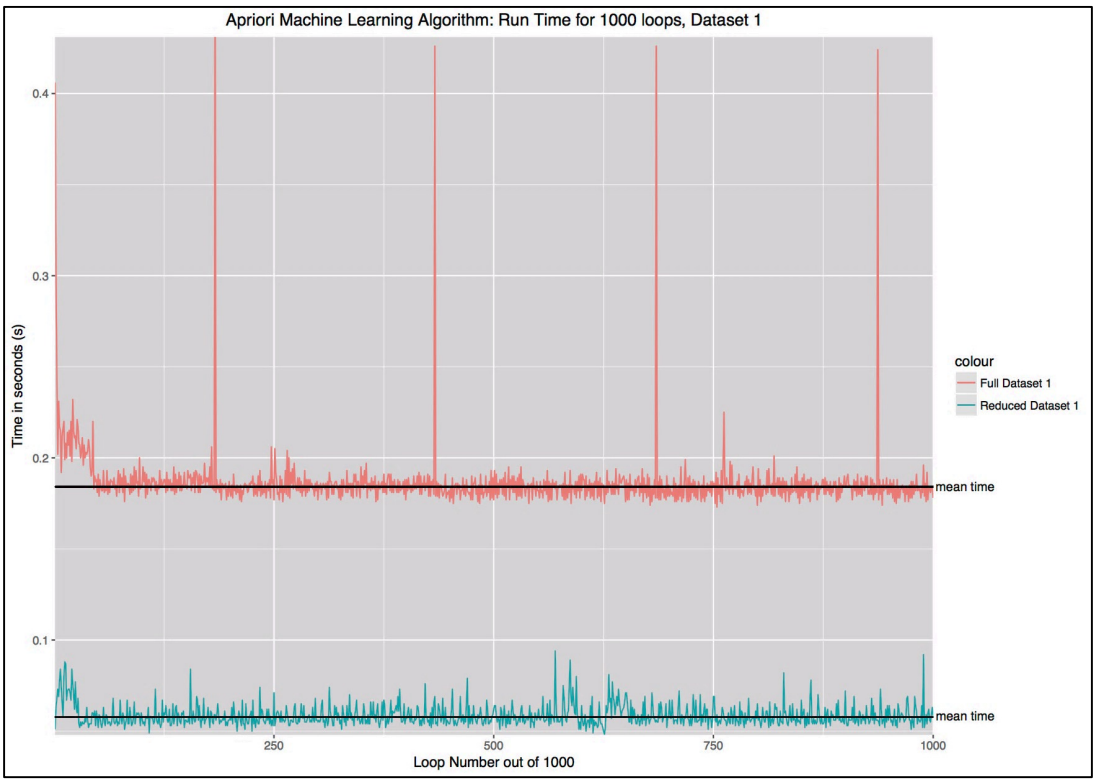


Figure 5.0.20, Illustrates the processing time in seconds(s) of the Apriori algorithm process RD1 and FD1. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

5.2.1.2 Apriori algorithm applied to Datasets Two

In this section, the results obtained from applying the Apriori algorithm to RD2 and FD2 are presented. Both datasets two have a total of 719 recorded sessions of adversary interactions with the longest sequence at 45 commands in length. The initial assessment conducted on a random 25% sample of FD2 showed the minimum support and confidence was to be set at 0.9 as stated in Table 5.7. From the pre-processing phase in Section 4.2.2, acquired dataset two had evidence of a script utilised by an adversary to interact with the Cowrie SSH honeypot consequently, there are similarities between RD2 and FD2. Yet, there was a 36.450% reduction in the number of unique commands from RD2 and FD2 by utilising the clustering technique mentioned in Section 4.2.2.4. However, when applying the Apriori algorithm to RD2 and FD2, the total number of extracted rules was 445,168 for both datasets as shown in Table 5.10 (responding *H2)*. As well as 240 unique rules identified after removing the duplicated rules, a reduction of 99.946% for both datasets.

Table 5.10, Shows the total number of rules extracted, number of unique rules after removing the duplicate rules and the percentage of duplicate rules extracted by the Apriori algorithm applied to RD2 and FD2.

Apriori algorithm	***FD2***	***RD2***
Total number of rules extracted	445,168	445,168
Number of unique rules after removing the duplicated rules	240	240
Percent of duplicate rules	99.946%	99.946%

Figure 5.21 and Figure 5.22 show the support and confidence of each of the unique 240 rules for RD2 and FD2. The lift is represented by the shade of each data point. Both Figure 5.21 and Figure 5.22 shows three clusters of rules, one cluster has a relatively low support and confidence values compared to the other clusters. There is also a cluster of rules with a low support value but high confidence value equating to 1. While the remaining cluster has a relative high support and high confidence value compared to the other clusters. Figures 5.21 and Figure 5.22 both depict, the unique rule set have an overall high support value, as reflected in the mean support of 0.936. Indicating the rules are consistent in both datasets. The confidence values for the unique rule sets are also high with the mean confidence value for RD2 and FD2 recorded at 1, signifying the commands within the rule sets are likely to appear together. The mean lift values for both datasets are 1.068, suggesting a dependence between the commands within the extracted rule sets. The compilation of the observations seen in Figure 5.21 and Figure 5.22 show comparable patterns can be extracted between the rules set between RD2 and FD2.

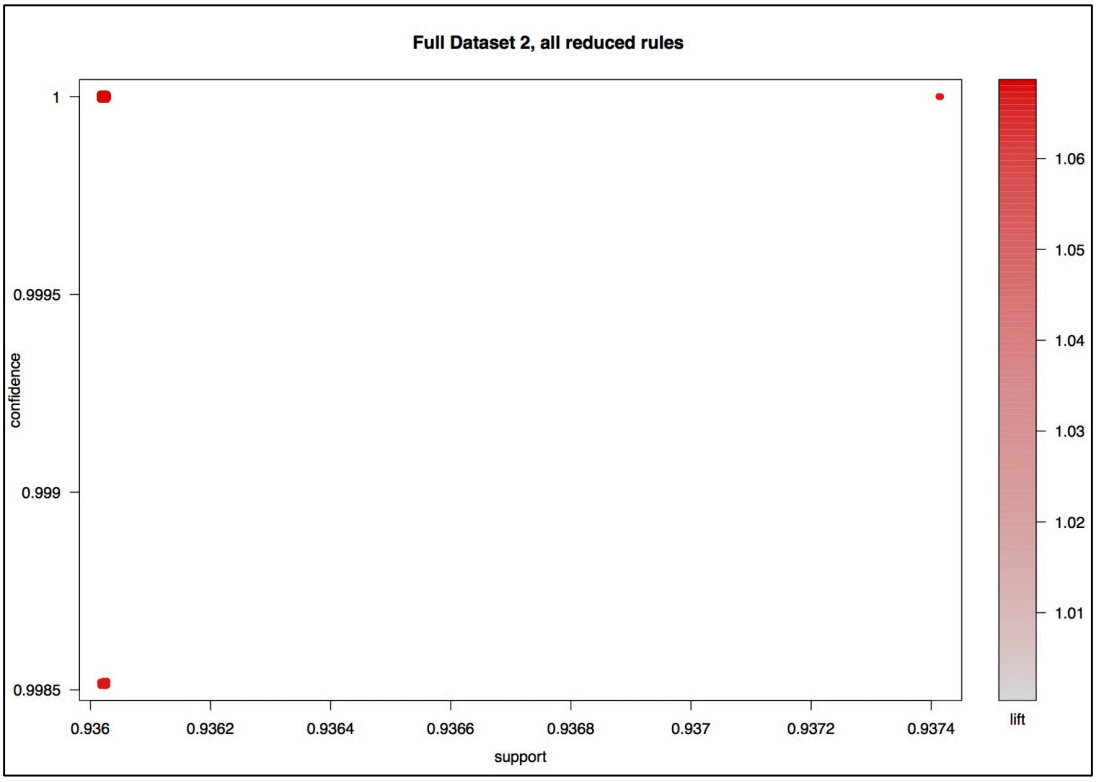


Figure 5.0.21, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Apriori algorithm from FD2.

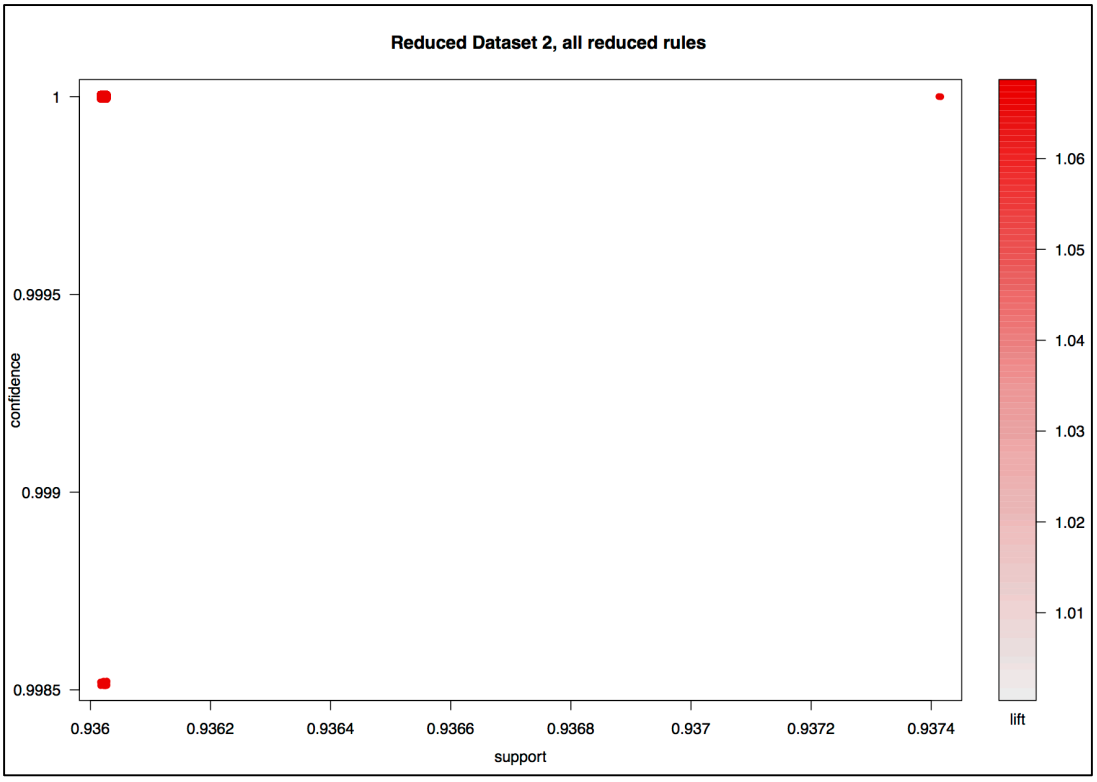


Figure 5.0.22, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Apriori algorithm from RD2.

To evaluate the precision of the extracted rules by the Apriori algorithm, the top 50 unique support based rules are examined (responding to *H4*). Figure 5.23 shows the support, associated confidence and lift values for the top 50 unique support based rules extracted from FD2. Figure 5.24 shows the top 50 rules for the unique support based rules extracted from RD2 and is similar to Figure 5.23. There is a likeness between the top 50 extracted rules displayed in Figure 5.21 and Figure 5.22 with three clusters of rules seen and those shown in Figure 5.23 and Figure 5.24. Table 5.11, shows the distribution of the top 50 unique support based rules from RD2 and FD2. The support, confidence and lift values are the same for RD2 as those of FD2. The distribution of the support values of the top 50 rules has a low spread from 0.936 as the minimum and a maximum of 0.937 as seen in Table 5.11. The low spread in the distribution of the top 50 rules also continuous for the confidence values, with a minimum of 0.999 and a maximum of 1, whilst the spread in the distribution of the lift is between 1.067 and 1.068.

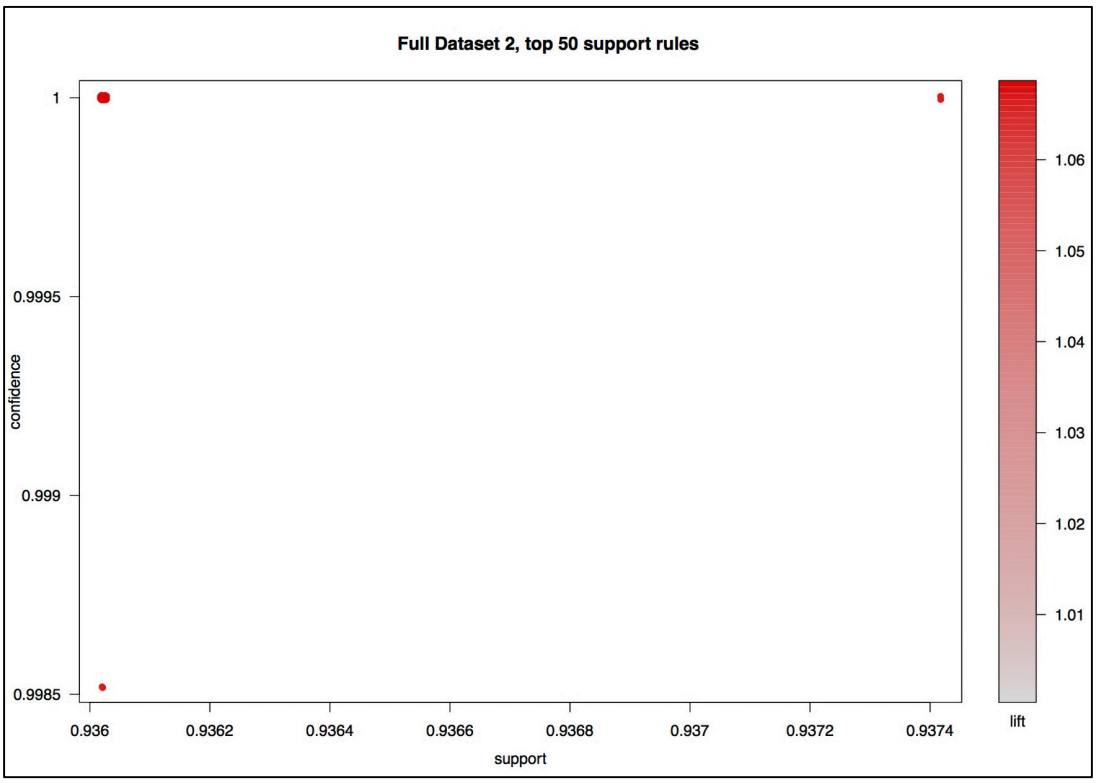


Figure 5.0.23, Illustrates a scatter plot of support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from FD2.

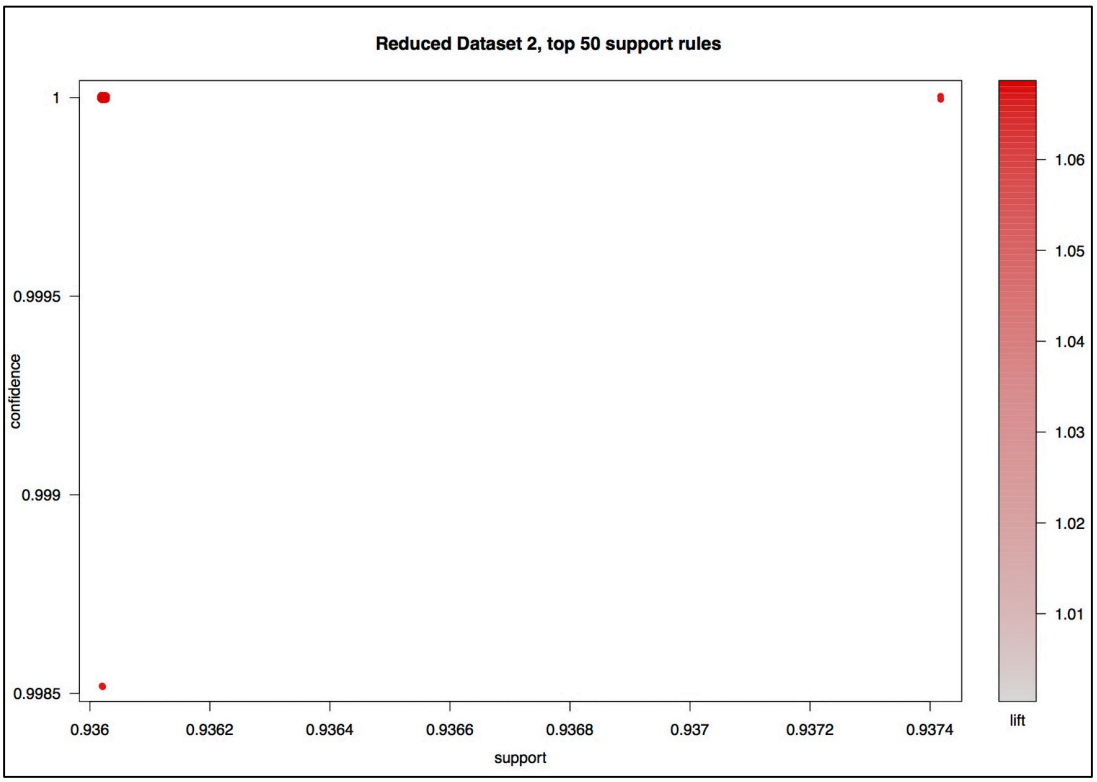


Figure 5.0.24, Illustrates a scatter plot of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from RD2.

Table 5.11, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Apriori algorithm applied to RD2 and FD2.

Apriori algorithm applied to the second datasets	***Support***		***Confidence***		***Lift***	
	Full	**Reduced**	**Full**	**Reduced**	**Full**	**Reduced**
Minimum	0.936	0.936	0.999	0.999	1.067	1.067
First quartile	0.936	0.936	1.000	1.000	1.068	1.068
Medium	0.936	0.936	1.000	1.000	1.068	1.068
Mean	0.936	0.936	1.000	1.000	1.068	1.068
Third quartile	0.936	0.936	1.000	1.000	1.068	1.068
Maximum	0.937	0.937	1.000	1.000	1.068	1.068

The similarities seen between RD2 and FD2 have also seen in the visual representation of the top 50 unique support based rules in Figure 5.25 and Figure 5.26. The arrows connecting the command and nodes represent the confidence between the extracted commands in a rule. The support is represented through the size of the nodes, the larger the node size the higher the associated support value. While the shade of the nodes symbolises the lift values of the commands within the extracted rule set, the darker the shade the higher the lift. As in previous figures, the top 50 support based rules for full datasets two illustrated in Figure 5.25 is similar to Figure 5.26 showing the top 50 rules for reduced datasets two. Both Figure 5.25 and Figure 5.26 illustrate 14 key interconnected commands in 16 different sequence placements forming the top 50 support based rule set. Suggesting the comparable patterns can be extracted from the rule set for RD2 and FD2.

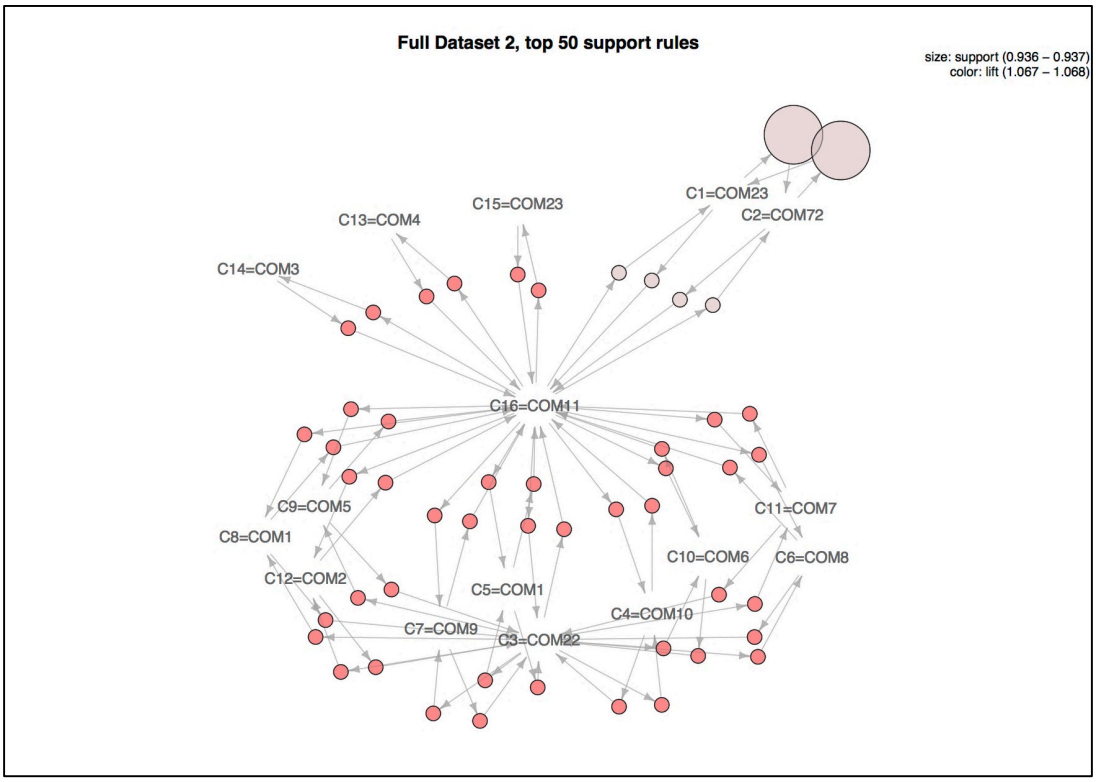


Figure 5.0.25, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from FD2.

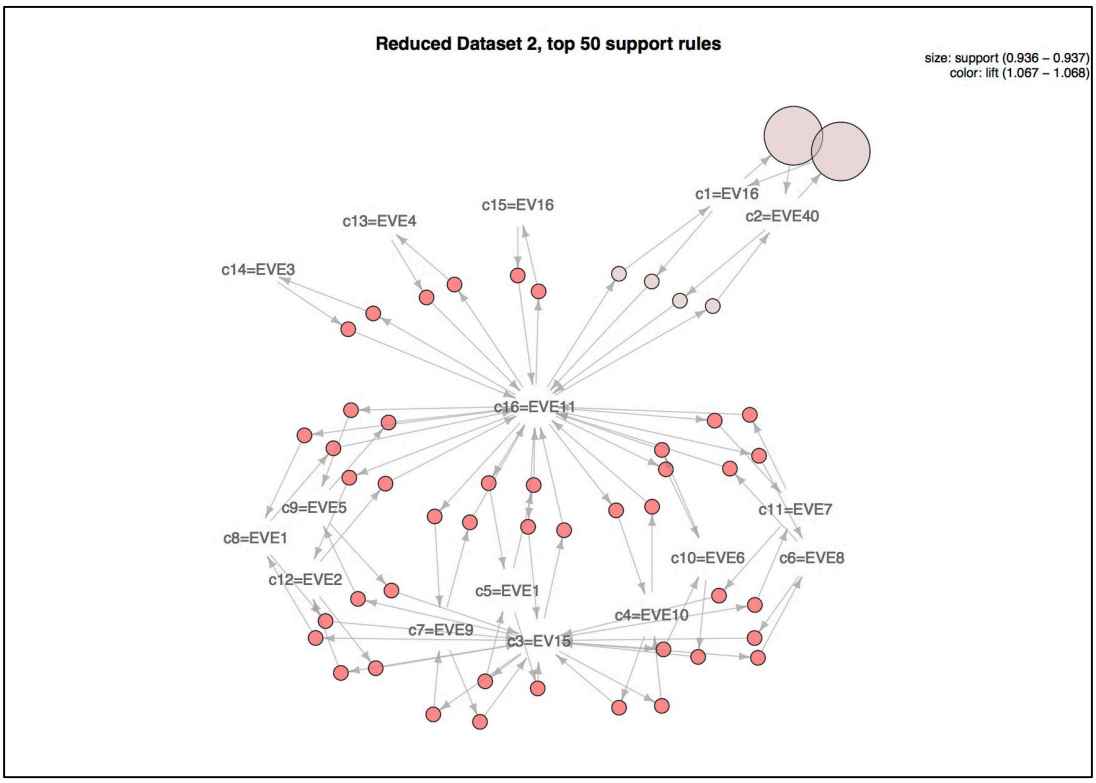


Figure 5.0.26, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from RD2.

The efficiency in seconds(s) of the Apriori algorithm to process RD2 and FD2 was also tested (responding to *H6*). As with the application of the Apriori algorithm on datasets one, the Apriori process was iterated 100 and 1,000 times. The black lines indicate the mean processing time after the outliers had been removed. Figure 5.27 shows the processing time for 100 iterations of the Apriori algorithm applied to RD2 and FD2. While Figure 5.28 showing the outcomes of 1,000 iterations of the Apriori algorithm applied to RD2 and FD2.

The processing time for RD2 is greater compared to FD2 as seen in Figure 5.27. Although the previous results of the extracted rules for RD2 and FD2 are similar, the efficacy of the Apriori algorithm to process the datasets are distinct. The greatest recorded processing time for FD2 was 5.346s while RD2 recorded 3.879s. With the lowest recorded processing time for FD2 recorded at 4.581s and 3.570s for RD2. Four outliers had been identified and removed from the results of the FD2 before calculating the mean processing time of 4.871s. While five outliers had been identified and extracted from the results from RD2 before calculating the mean processing time of 3.798s. The black lines in Figure 5.27 represent the mean processing time for both RD2 and FD2. Indicating the Apriori algorithm is on average 1.073s efficient at processing RD2 compared to FD2.

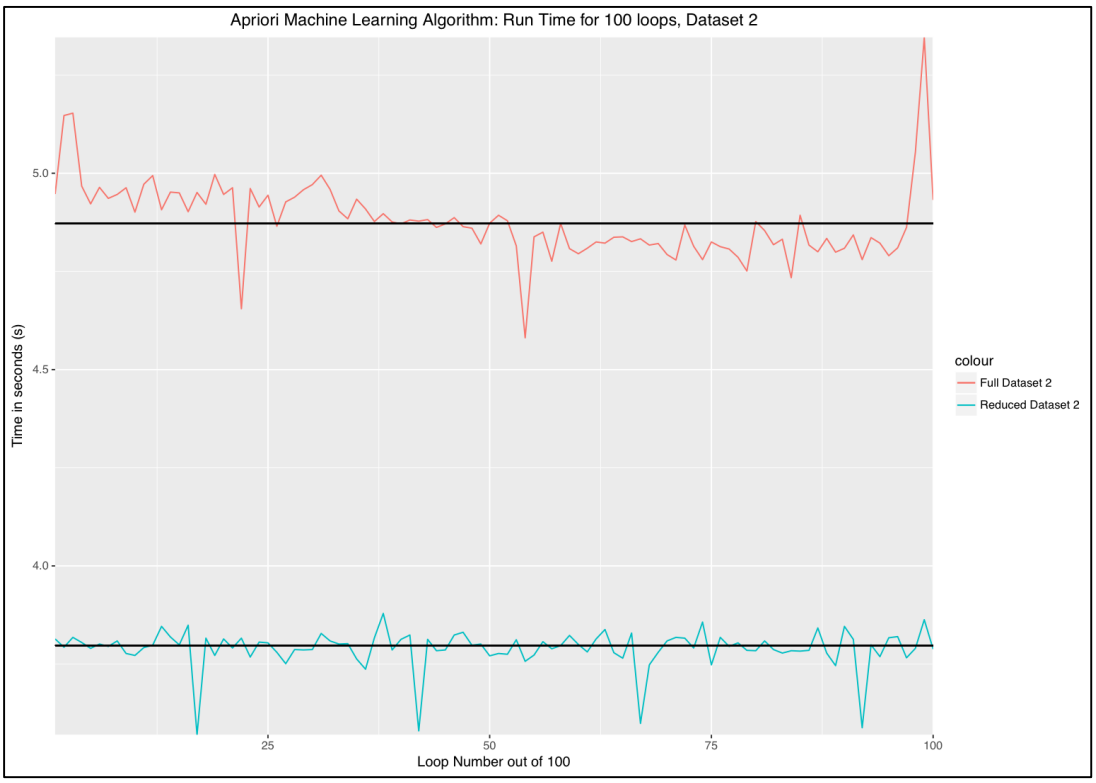


Figure 5.0.27, Illustrates the processing time in seconds(s) of the Apriori algorithm process RD2 and FD2. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

Figure 5.28 depicts the results of 1,000 iterations of the application of the Apriori algorithm to process RD2 and FD2. Similar to Figure 5.27, FD2 has a greater processing time compared to RD2, as seen in Figure 5.28. The greatest recorded processing time for FD2 using the Apriori algorithm was 8.111s. While the RD2 recorded 4.561s as the greatest recorded processing time. The lowest processing time for FD2 is 4.358s and 3.475s for RD2. The mean processing time for FD2 was calculated after 66 outliners had been removed resulting in the mean value of 4.9236s represented as a black line in Figure 5.28. A total of 143 outliers had been removed from RD2 and a mean value of 3.786s was calculated. There is a difference of 1.137s between both means, indicating the Apriori algorithm is efficient at processing RD2. The aggregated results are presented in Section 5.3 along with the corresponding hypotheses tested. In the following section the results from applying the Apriori algorithm to RD3 and FD3.

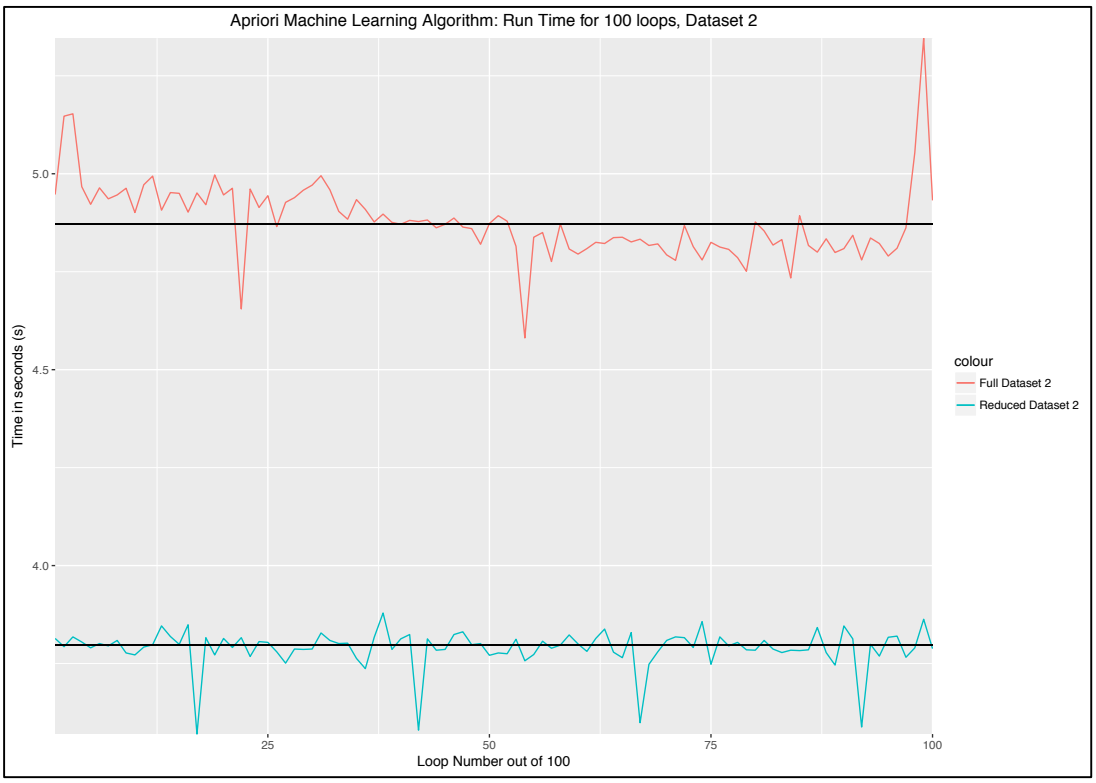


Figure 5.0.28, Illustrates the processing time in seconds(s) of the Apriori algorithm process the RD2 and FD2. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

5.2.1.3 Apriori algorithm applied to Datasets Three

Within this section, the results of applying the Apriori algorithm to RD3 and FD3 are presented below. A total of 318 sessions of adversary interactions had been recorded for datasets three, with the longest sequence at 220 commands in length. Based on the initial assessment conducted on a random 25% sample of FD3, 0.01 was the minimum support and confidence parameter values to be set as mentioned in Table 5.7. The pre-processing phase in Section 4.2.3 showed FD3 had a total of 748 unique commands identified after applying the clustering technique, RD3 contained 338 unique commands, that is a reduction of 45.19%. Table 5.12 shows the total number of rules extracted using the Apriori algorithm for FD3 was 526 while a total of 568 rules had been extracted from RD3 (responding to *H2*). After the duplicated rules had been removed 125 unique rules remained for FD3, a reduction of 76.236%. Compared to 170 unique rules remaining for RD3 a reduction of 70.070%. The increased number of total rules extracted and the number of unique rules for RD3 compared to FD3, could suggest addition patterns have been extracted.

Table 5.12, Shows the total number of rules extracted, number of unique rules after removing the duplicate rules and the percentage of duplicate rules extracted by the Apriori algorithm applied to RD3 and FD3.

Apriori algorithm	***FD3***	***RD3***
Total number of rules extracted	526	568
Number of unique rules after removing the duplicated rules	125	170
Percent of duplicate rules	76.236%	70.070%

The 125 unique rules extracted from FD3 are depicted in Figure 5.29 and the 170 unique rules extracted from RD3 are shown in Figure 5.30. Figure 5.29 and Figure 5.30 both show RD3 and FD3 have an overall low support value, with the majority of the rules having a support value between 0.01 and 0.02. The unique rule set extracted from FD3 have a mean support value of 0.016. While a mean support value of 0.016 was recorded for the unique rule set extracted from RD3. There are two rules that have a support value of around 0.09 and a low associated lift value seen in both Figure 5.29 and Figure 5.30. The mean lift value for FD3 was 30.659 and the mean lift value for RD3 was 23.685. Although the unique rule sets have a low support rate the confidence value ranges between 0 and 1. As seen in Figure 5.29 and Figure 5.30 the rules that have a confidence value of 1 have a low support rate of 0.01. The mean confidence value for the unique rule set extracted from FD3 was 0.616, while RD3 had a mean support value of 0.572 for the unique rule set extracted.

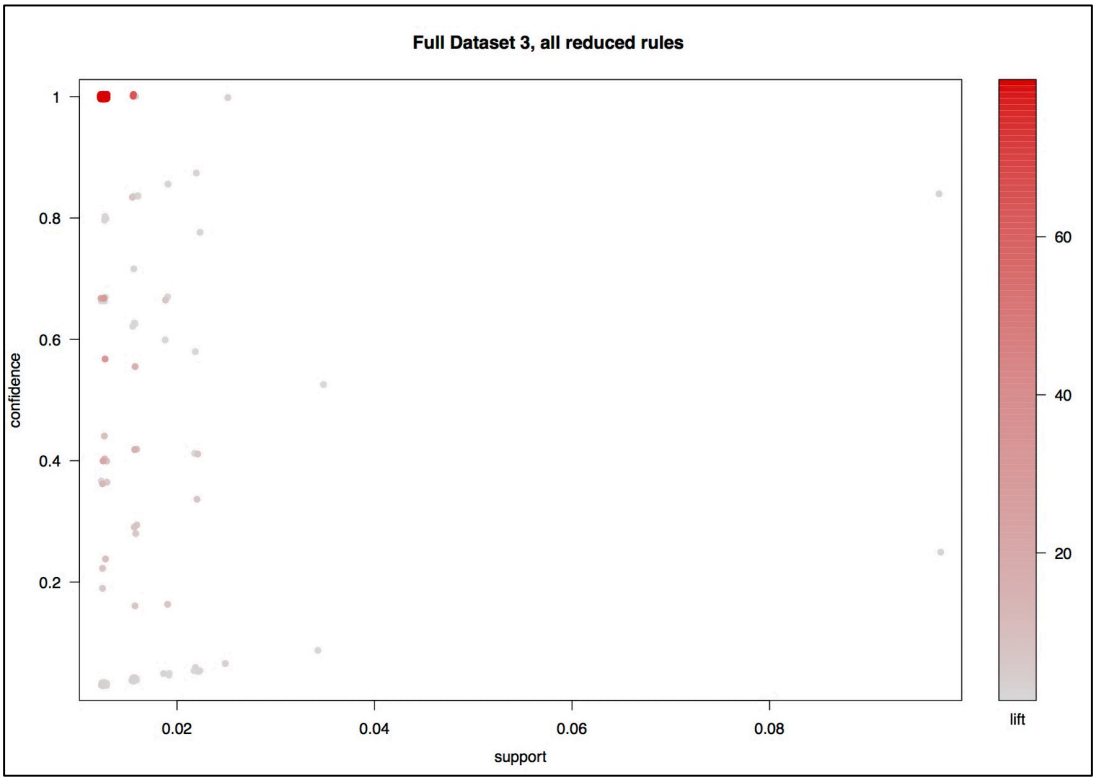


Figure 5.0.29, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Apriori algorithm from FD3.

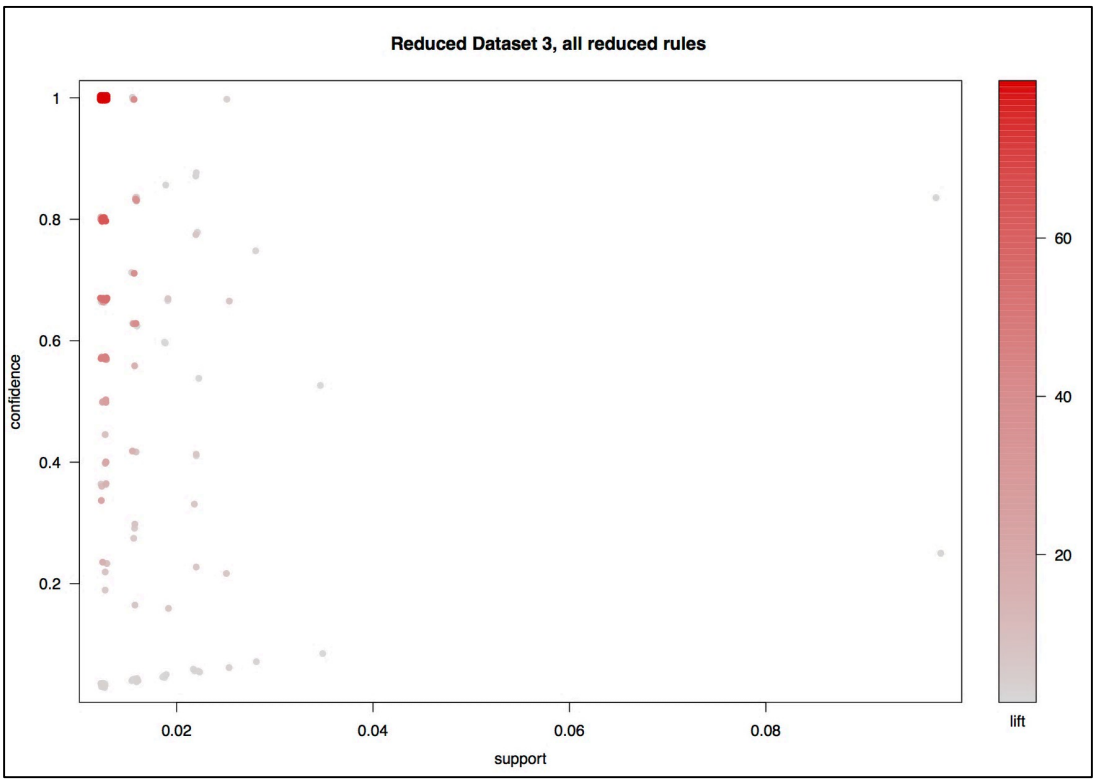


Figure 5.0.30, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Apriori algorithm from RD3.

The precision of the rule sets extracted by the Apriori algorithm from RD3 and FD3 evaluated based on the top 50 unique support based rules. The support value was selected as it measures the probability of the rules occurring in the datasets. Figure 5.31 depicts the top 50 unique support based rules of FD3. Figure 5.32 illustrates the top 50 unique support based rules for RD3. Figure 5.31 shows the extracted rules have fewer rules with a greater lift value compared to Figure 5.32. Figure 5.32 has four data points with a greater associated lift value, while FD3 depicted in Figure 5.31 has two data points with a greater associated lift value. The lift value represents the dependence between the commands in the rules, the lower the lift the more dependence there is between the commands. Additionally, as seen in Figure 5.29 and Figure 5.30 the confidence for the rules range between 0 and 1, while the support values for the extracted rule sets shown in Figure 5.31 and Figure 5.32 predominantly are between 0.01 and 0.04. The precision of the extracted rules are based on the distribution of the of rule sets provided in Table 5.13.

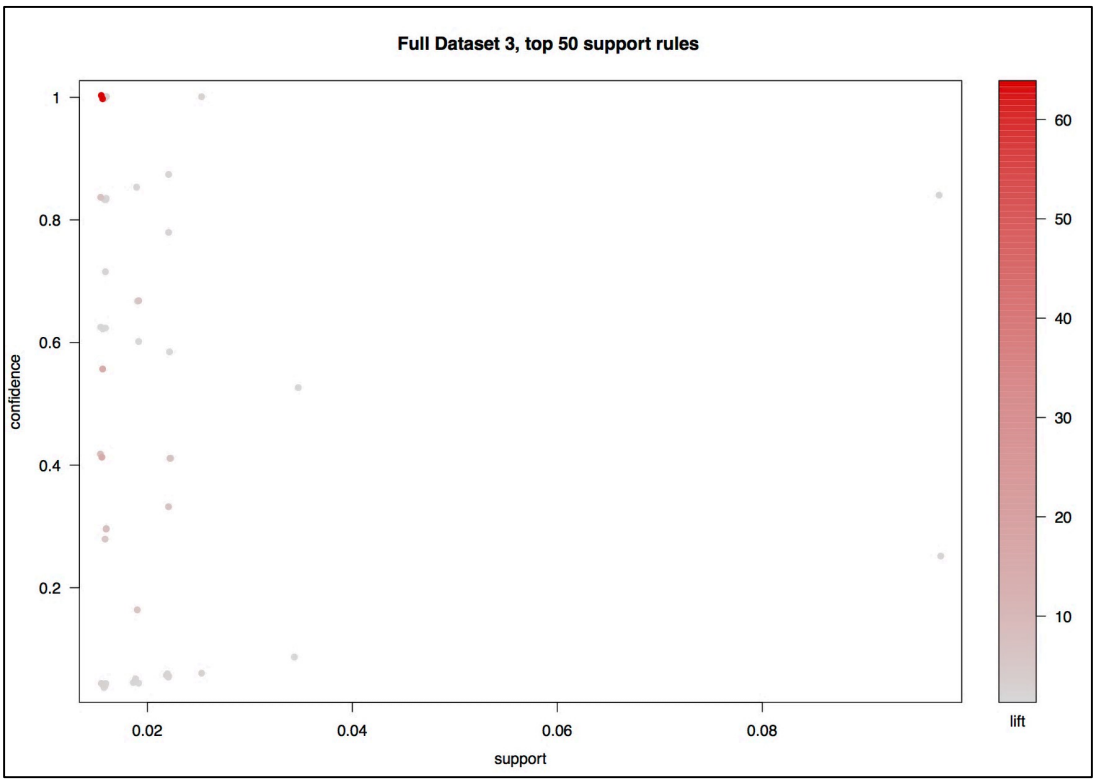


Figure 5.0.31, Illustrates a scatter plot of support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from FD3.

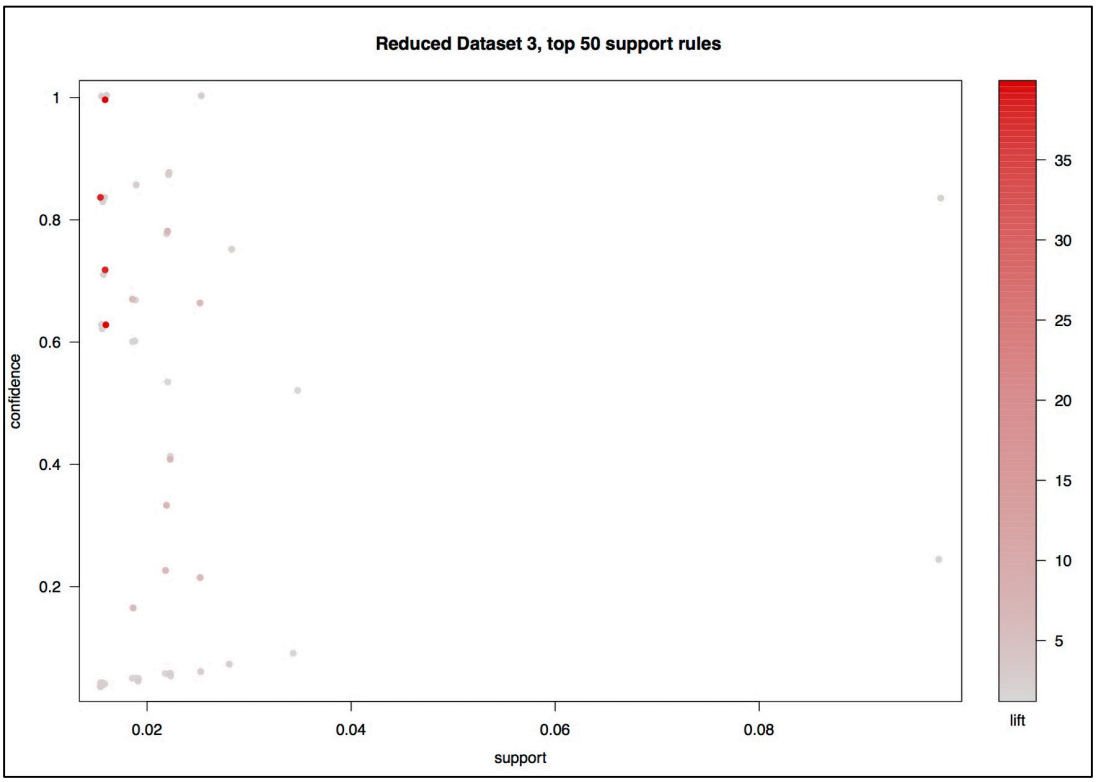


Figure 5.0.32, Illustrates a scatter plot of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from RD3.

The distribution of the top 50 unique support based rule sets for RD3 and FD3 are presented in Table 5.13 (responding to *H4*). The distribution of the rule sets are used to evaluate the precision of the extracted rules using the Apriori algorithm. The support distribution between RD3 and FD3 are similar, with the same minimum, first quartile, third quartile and the maximum. However, the medium support value of 0.019 for RD3 is greater compared to the medium support value of 0.015 for FD3. The mean support value is also greater for RD3 at 0.023, compared to the mean support value of 0.022 for FD3. Consequently, the trend also continues in the distribution of the confidence values, with RD3 having a greater medium, mean as well as third quartile value compared to the corresponding values for FD3. Apart from the mean, the third quartile and maximum lift vales had been the same for RD3 and FD3. RD3 has a lower mean lift value at 5.499 compared to 5.679 for FD3. As well as a lower third quartile lift value at 2.544 for RD3 compared to 4.533 for FD3.

Table 5.13, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Apriori algorithm applied to RD3 and FD3.

Apriori algorithm applied to the third datasets	***Support***		***Confidence***		***Lift***	
	Full	**Reduced**	**Full**	**Reduced**	**Full**	**Reduced**
Minimum	0.015	0.015	0.040	0.040	1.048	1.048
First quartile	0.015	0.015	0.056	0.056	1.590	1.590
Medium	0.015	0.019	0.414	0.531	2.126	2.126
Mean	0.022	0.023	0.440	0.454	5.679	5.499
Third quartile	0.022	0.022	0.762	0.770	4.533	2.544
Maximum	0.097	0.097	1.000	1.000	63.600	39.750

Figure 5.33 is a visual representation of the top 50 unique support based rules for FD3 and Figure 5.34 is the visual representation for RD3. Figure 5.33 depicts a single cluster of rules, while Figure 5.34 has two clusters of rules and a central cluster. The confidence values of the rule sets are represented by the arrows connecting the commands along with the nodes and the support values are represented by the size of the node, the greater the support the larger the node. The shade of the nodes corresponds to the lift value of the rules. Figure 5.33 illustrates FD3 with 12 key interconnected commands in 23 different sequence placements. While Figure 5.34 depicts the top 50 unique support based rules for RD3, with 14 key interconnected commands in 24 different sequence placements. Indicating, additional patterns can be extracted from the rule set between RD3 and FD3.

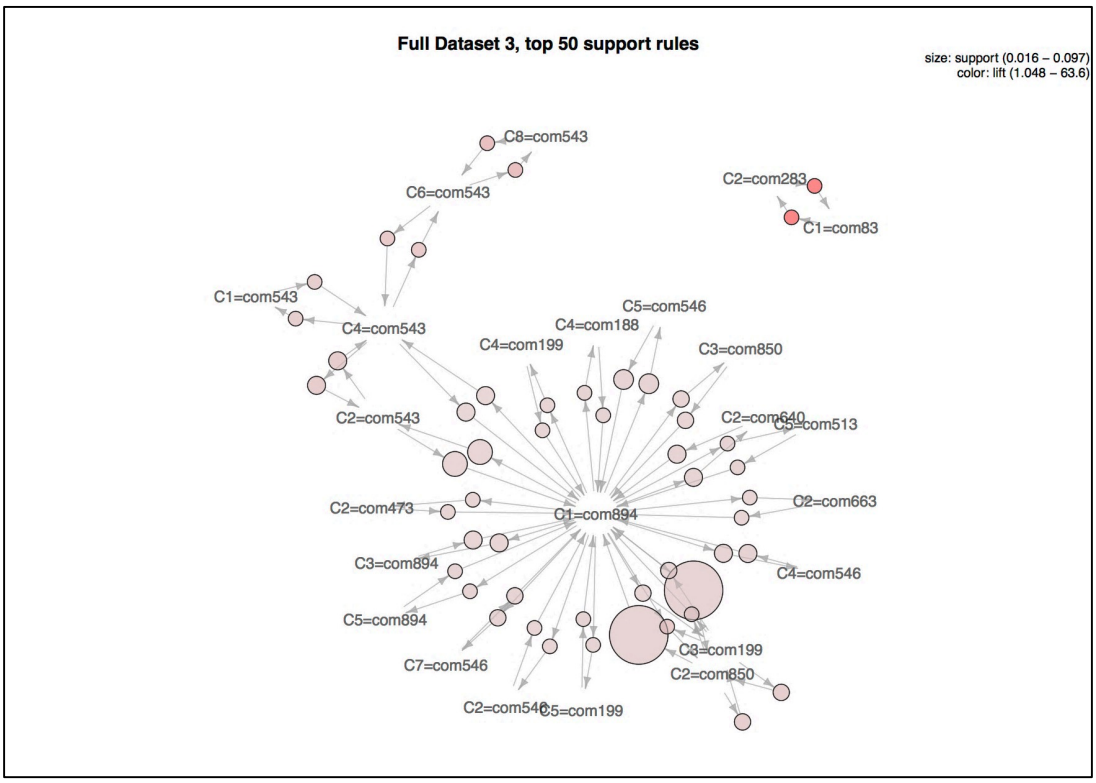


Figure 5.0.33, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from FD3.

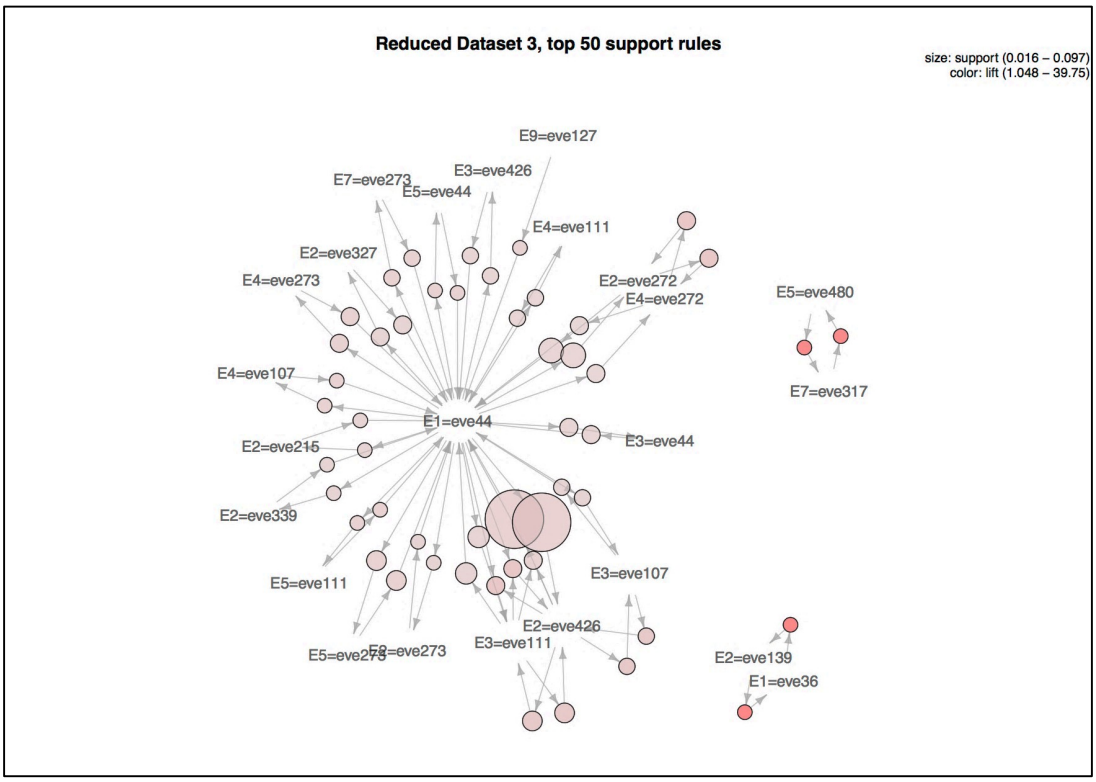


Figure 5.0.34, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Apriori algorithm from RD3.

The efficiency of the Apriori algorithm was verified by applying the algorithm to RD3 and FD3 iterated 100 and 1,000 times (responding to *H6*). Figure 5.35 shows results of the Apriori algorithm iterated 100 times and Figure 5.36 shows the application of the algorithm iterated 1,000 times. The black link shows the mean for the RD3 and FD3 after the outliers have been removed.

From Figure 5.35 the Apriori algorithm is efficient at processing RD3 and FD3. The greatest recorded process time for FD3 was 1.044s and the lowest recorded process time for RD3 was 0.475s. While the greatest recorded process time for RD3 was 0.897s and the lowest recorded process time was 0.581s as seen in Figure 5.35. After removing 14 outliers the mean processing time for FD3 was calculated at 0.488s. Additionally, a total of 9 outliers had been removed before the mean value of 0.596s was calculated for RD3. The mean values are represented by the black lines in Figure 5.35. The data presented in Figure 5.35 suggest the Apriori algorithm is efficient in processing FD3 compared to RD3.

Figure 5.36 shows FD3 is efficient compared to RD3 as the Apriori algorithm was applied to the datasets. Figure 5.36 displays the same outcome as Figure 5.35. However, there are incremental data points that have a greater processing time. The greatest recorded processing time recorded for FD3 was 0.771s and 1.041s was the greatest recorded processing time for RD3. In Figure 5.36, the lowest recorded processing time for FD3 was 0.471s and 0.577s was the greatest recorded processing time for RD3. A total of 50 outliers had been removed from both datasets. The mean processing time for FD3 is presented by the black link is 0.4896s and the mean processing time for RD3 is 0.597s. The aggregated results are presented in Section 5.3 along with the corresponding hypotheses tested. In the following section the results from applying the Eclat algorithm to the three reduced datasets and their respective full datasets.

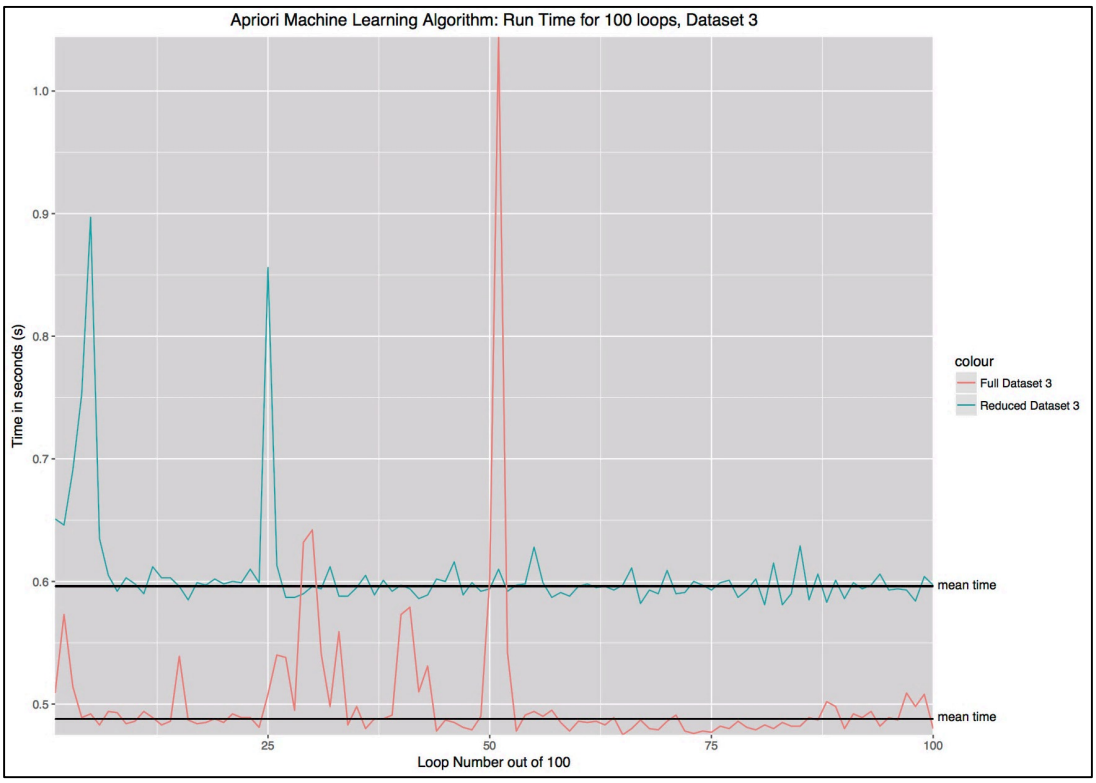


Figure 5.0.35, Illustrates the processing time in seconds(s) of the Apriori algorithm process RD3 and FD3. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

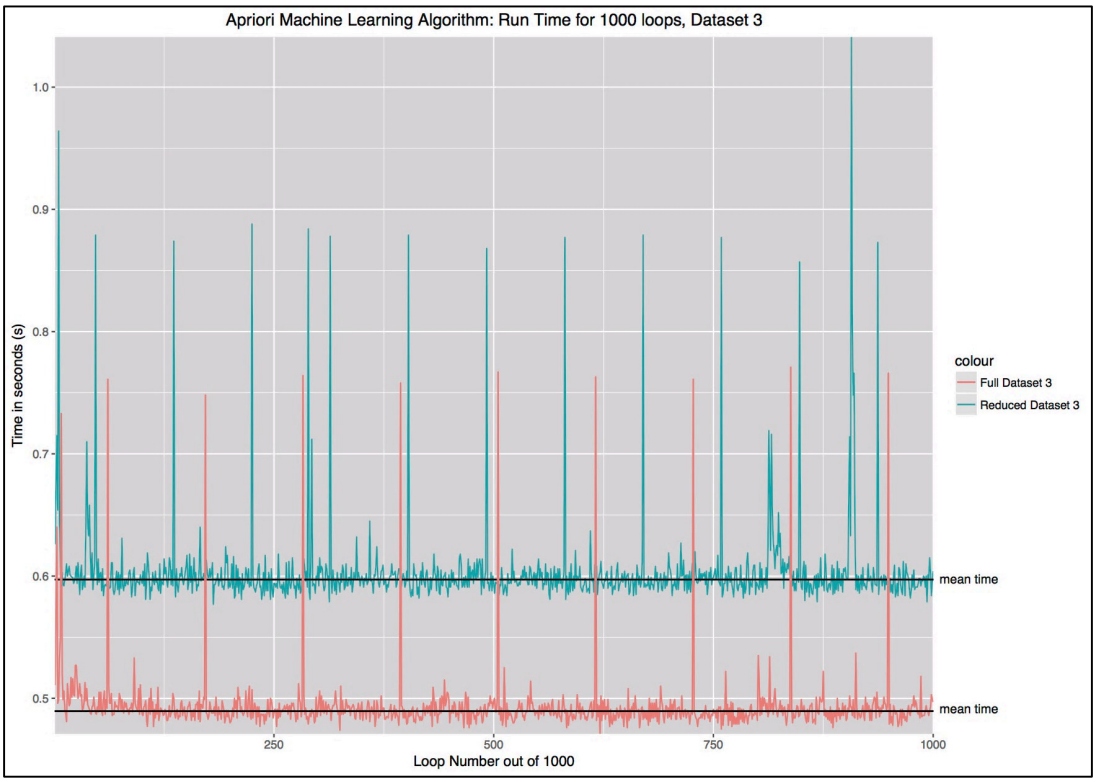


Figure 5.0.36, Illustrates the processing time in seconds(s) of the Apriori algorithm process RD3 and FD3. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

5.2.2 Eclat algorithm

Eclat is an association rule mining algorithm similar to the Apriori algorithm. The Eclat algorithm focuses on the frequency of objects occurring within a given dataset, by utilising a depth-first search. The algorithm is typically efficient at processing a given dataset compared to the Apriori algorithm. The *R* package arules implementation of the Eclat algorithm was used for this study. The support parameters used for the Apriori algorithm presented in Table 5.7 are also used for the application of Eclat algorithm. As the Eclat algorithm extracts item sets are based on the support parameter, before formulating the rule sets.

With the Eclat algorithm, frequent objects are extracted before the rules are formulated. The extracted rules have been used to determine whether additional patterns had been extracted from the rule sets for reduced datasets and their respective full datasets (responding to *H2*). The precision of the extracted rule sets are evaluated using the top 50 formulated rules based on the support value, by comparing the distribution of the rule sets (responding to *H4*). The efficiency of the Eclat algorithm to process the datasets are measured by iterating the process 100 and 1,000 times (responding to *H6*). Thereafter, the outliners had been removed to identify the mean processing time for the Eclat algorithm to process the reduced datasets and their respective full datasets.

5.2.2.1 Eclat algorithm applied to Datasets One

In this section, the results of applying the Eclat algorithm to RD1 and FD1 are presented. Datasets one had a total of 1,128 recorded sessions of adversary interactions recorded. The pre-processing phase in Section 4.2.1 of dataset one showed FD1 had a total of 2,960 unique commands. While the clustering process resulted in RD1 containing a total of 406 unique commands. The initial assessment for datasets two suggests the minimum support value should be set to 0.05 as stated in Table 5.7. Unlike the Apriori algorithm, where the rules extracted from a dataset before removing the duplicate rules from the rule set. The Eclat algorithm extracts item sets, these are sets of frequent items within the given dataset before the rule sets are formulated then the duplicate rules are removed.

The Eclat algorithm extracted 259 item sets from FD1 and formulating 977 rules as seen in Table 5.14 (responding *H2)*. Item sets are extracted are frequent command sets occurring within the given dataset. The rules are formulated based on the extracted item sets before the duplicate rules are removed. As shown in Table 5.14, a total of 59,201 item sets had been extracted from RD1 and 446,465 rules had been formulated. Upon removing the duplicated rules, 64 unique rules remain for FD1 and 1,441 unique rules remain for RD1 (responding to *H2*). The unique rules extracted by the Eclat algorithm are less than those extracted by the Apriori algorithm, as shown in Table 5.8. The support, confidence and lift

of the unique rules extracted by the Eclat algorithm applied to FD3 in Figure 5.37 and RD3 in Figure 5.38.

Table 5.14, Shows the total number of rules extracted, number of unique rules after removing the duplicate rules and the percentage of duplicate rules extracted by the Eclat algorithm applied to RD and FD1.

Eclat algorithm	***FD1***	***RD1***
Total number of item sets extracted	259	59,201
Number rules formulated	977	446,465
Number of unique rules after removing the duplicated rules	64	1,441

Figure 5.37 shows the scatter plot of the unique rules by the Eclat algorithm on FD1, a total of 64 rules are depicted. The support and associated confidence and lift of the extracted rules are displayed in Figure 5.37. Predominantly the rules have a low support around 0.07 but these rules have a confidence value ranging between 0.97 and 1. The lift values vary, with a cluster of rules with a low support value and greater confidence value, having an associated increased lift value. The increase in the lift value signifies the commands in the extracted rules are likely to be independent. Figure 5.38 shows the 1,441 unique rules extracted from the Eclat algorithm applied to RD1. There is a cluster of rules with a support rate between 0.06 and 0.1 with the greater confidence value between 0.8 and 1. Figure 5.38 shows the increase in the number of unique rules extracted between RD1 and FD1. The mean support value was 0.072 for Figure 5.37 and 0.992 as the mean confidence value. Figure 5.38 has a lower mean support value of 0.070 compared to Figure 5.37 as well as a lower mean confidence value of 0.953. However, the mean lift value of the formulated rules for FD1 was 11.451 while the mean lift value for RD1 was 9.802. In order the evaluate the precision of the rules extracted, the top 50 support based rules are evaluated in the following sections.

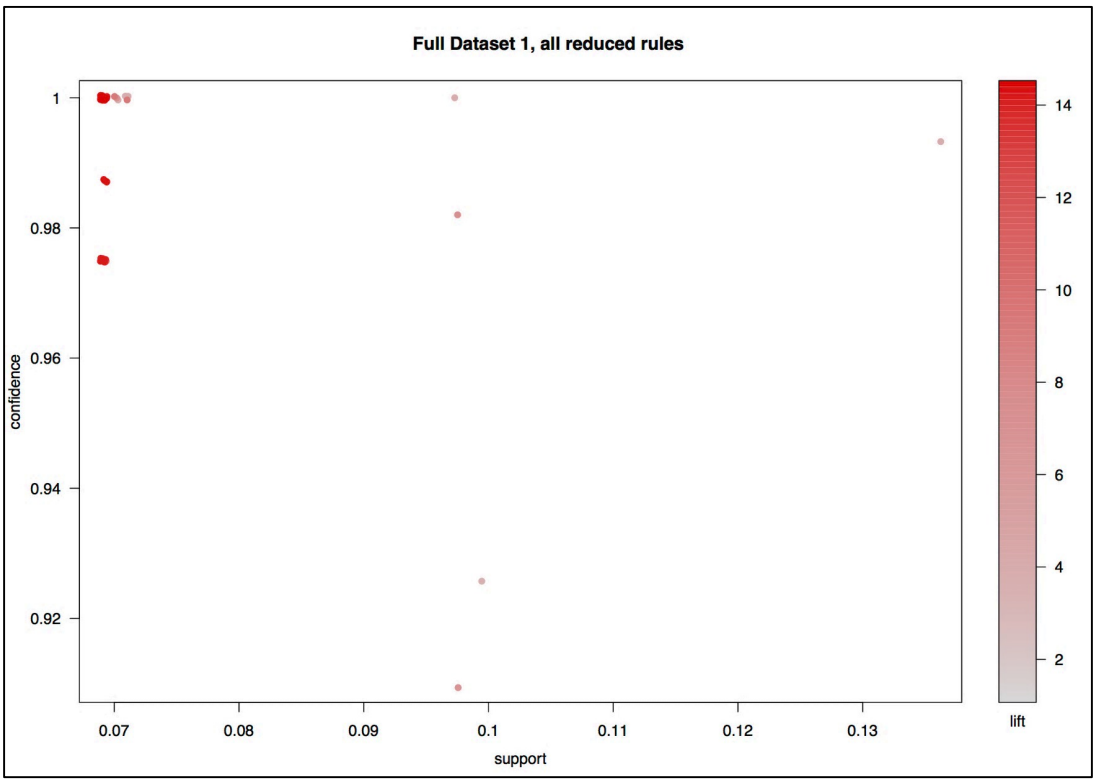


Figure 5.0.37, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Eclat algorithm from FD1.

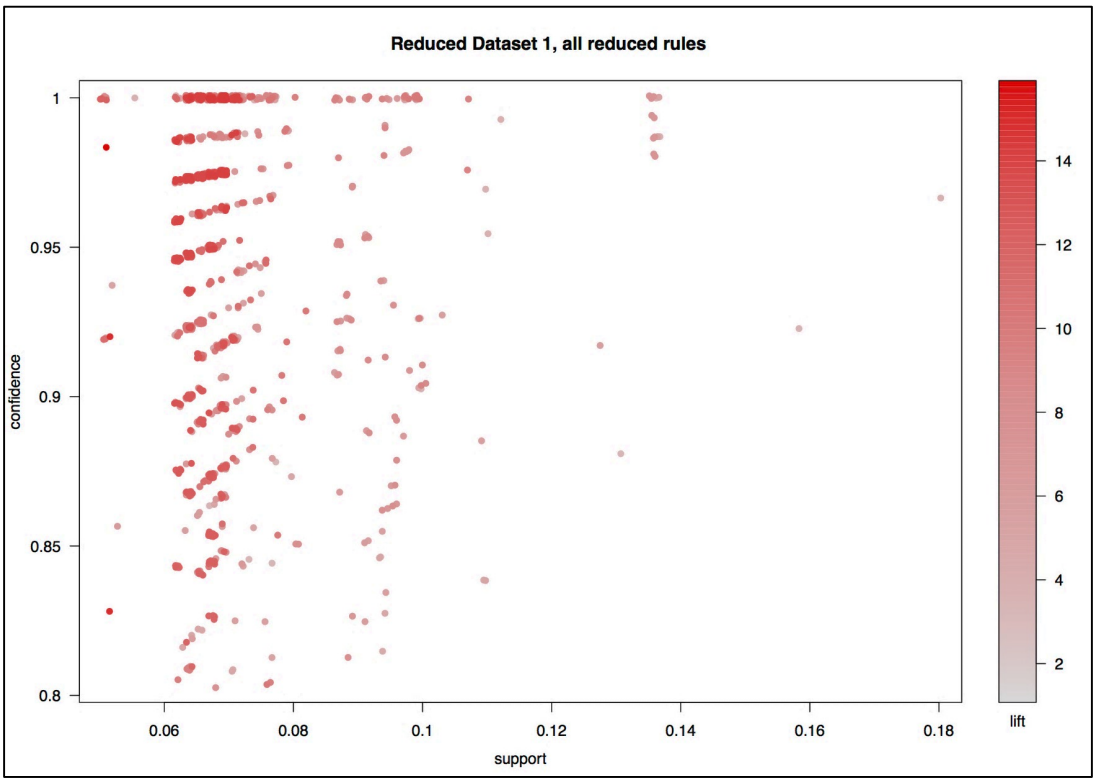


Figure 5.0.38, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Eclat algorithm from RD1.

The precision of the unique extracted rules is evaluated using the top 50 support based rules. Figure 5.39 shows the top 50 support based rules for FD1, the support of the rule sets is predominantly low at 0.07, while the confidence values ranging from 0.96 to 1. There is a single data point that had a support value exceeding 0.13. On the other hand, Figure 5.40 shows the top 50 support based rules of RD1. The support values predominantly range between 0.1 and 0.14 and the confidence value ranges between 0.85 and 1. Figure 5.39 shows the rules are scattered compared to that of Figure 5.40. The distribution of the top 50 unique support based rule sets are presented in Table 5.15, that shows the minimum, first quartile, medium, mean, third quartile and maximum value for the support, confidence and lift parameters.

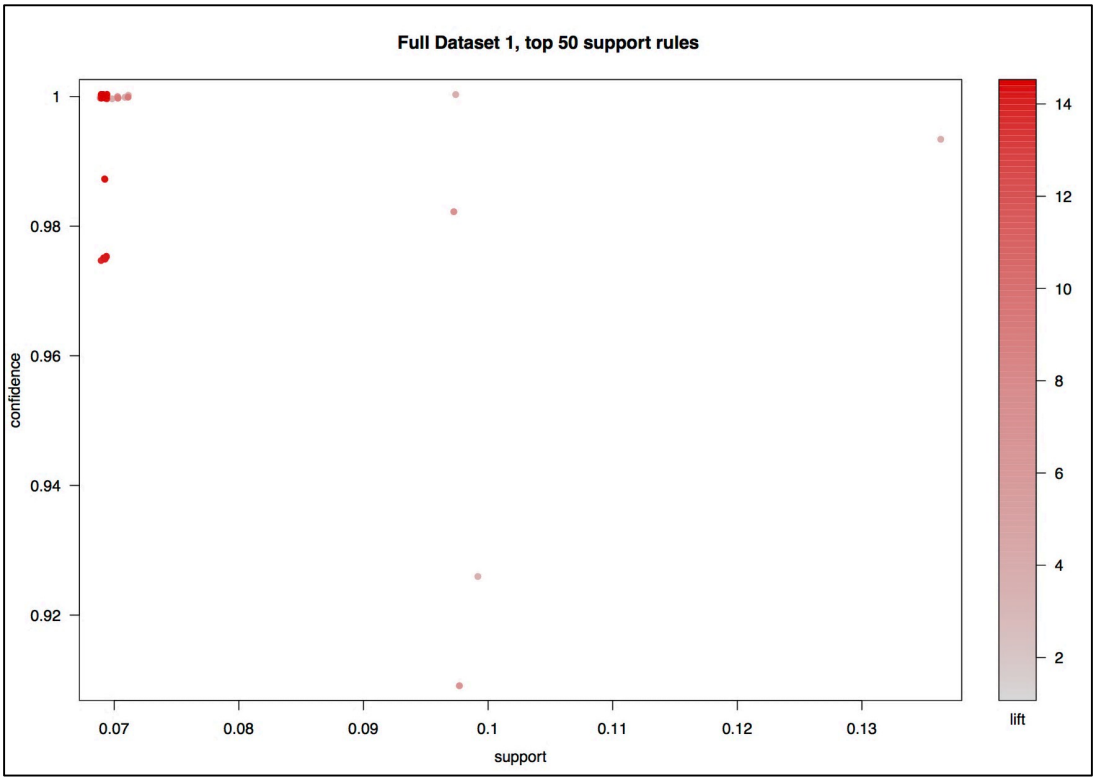


Figure 5.0.39, Illustrates a scatter plot of support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from FD1.

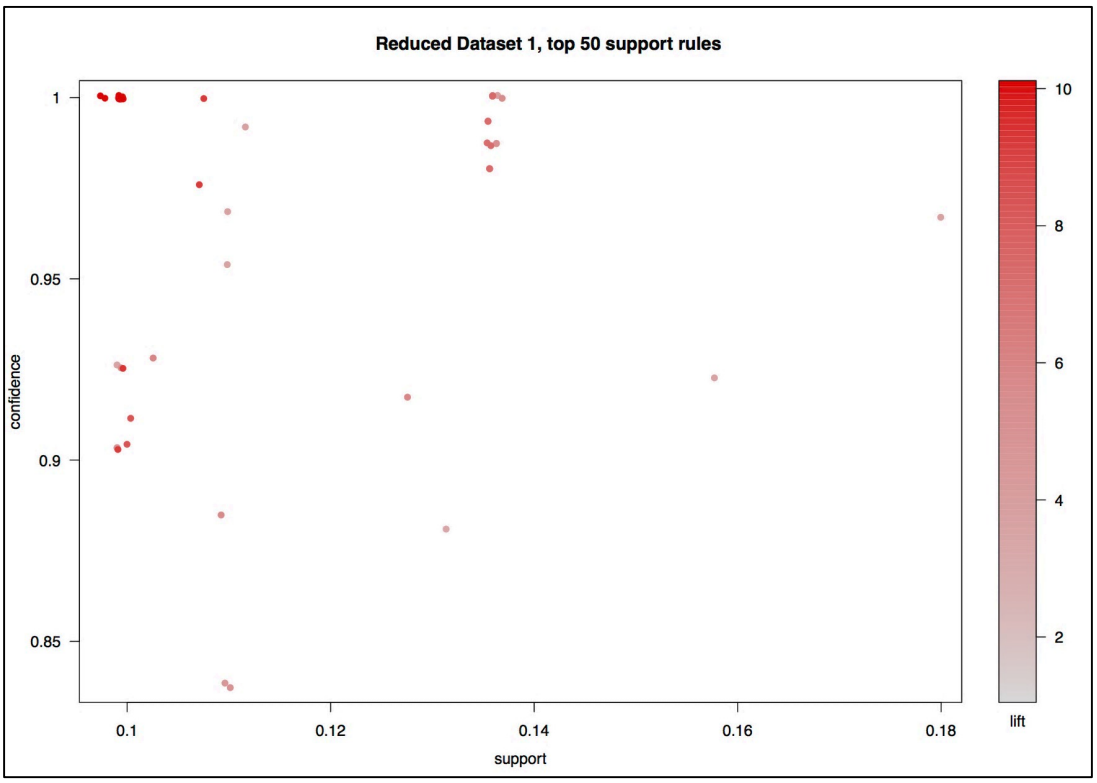


Figure 5.0.40, Illustrates a scatter plot of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from RD1.

The distribution of the top 50 support based rules formulated using the Eclat algorithm applied to RD1 and FD1 are presented in Table 5.15 (responding *H4)*. The support distribution of RD1 is greater than the support distribution for FD1. The mean support value for RD1 is 0.070 while the mean support value for FD1 was greater at 0.072. The support values represent the applicability of the unique rule sets to the given dataset. The maximum support value was the only value greater for RD1 at 0.180 compared to FD1 at 0.137. The mean confidence value of 0.953 was recorded for RD1 and is lower compared to the mean confidence value of 0.992 recorded for FD1. The confidence value for a rule suggests the commands in the rule set are likely to appear together within the given dataset. As for the lift values seen in Table 5.15, the RD1 has a greater distributed value compared to the lift distribution of FD1. However, the mean lift value for FD1 was greater at 11.451 compared to the mean lift value of 9.802 for RD1. A lower lift value suggests the commands in the rule sets are dependent on each other. A visual representation of the top 50 support based rules are presented in Figure 5.41 for FD1 and Figure 5.42 illustrates the top 50 support based for RD1.

Table 5.15, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Eclat algorithm applied to RD1 and FD1.

Eclat algorithm applied to the first datasets	***Support***		***Confidence***		***Lift***	
	Full	**Reduced**	**Full**	**Reduced**	**Full**	**Reduced**
Minimum	0.069	0.051	0.909	0.802	3.469	3.165
First quartile	0.069	0.064	0.987	0.923	7.277	7.185
Medium	0.069	0.067	1.000	0.973	14.100	10.071
Mean	0.072	0.070	0.992	0.953	11.451	9.802
Third quartile	0.069	0.069	1.000	1.000	14.278	12.357
Maximum	0.137	0.180	1.000	1.000	14.462	15.841

Figure 5.41 is a visual representation of the top 50 support base rules extracted by the Eclat algorithm from full data one. The confidence of the rule sets is represented by the arrows connecting the nodes and commands together. The support values are represented by the size of the nodes, the large the node the greater the support value, while the lift is presented by the colour of the node the darker the shade the greater the associated lift. The rules in Figure 5.41 are organised into a single cluster. There are four key interconnected commands, in eight different key placements. While as seen in Figure 5.42 there three clusters of rules, with 10 key interconnected commands, in 16 different key placements.

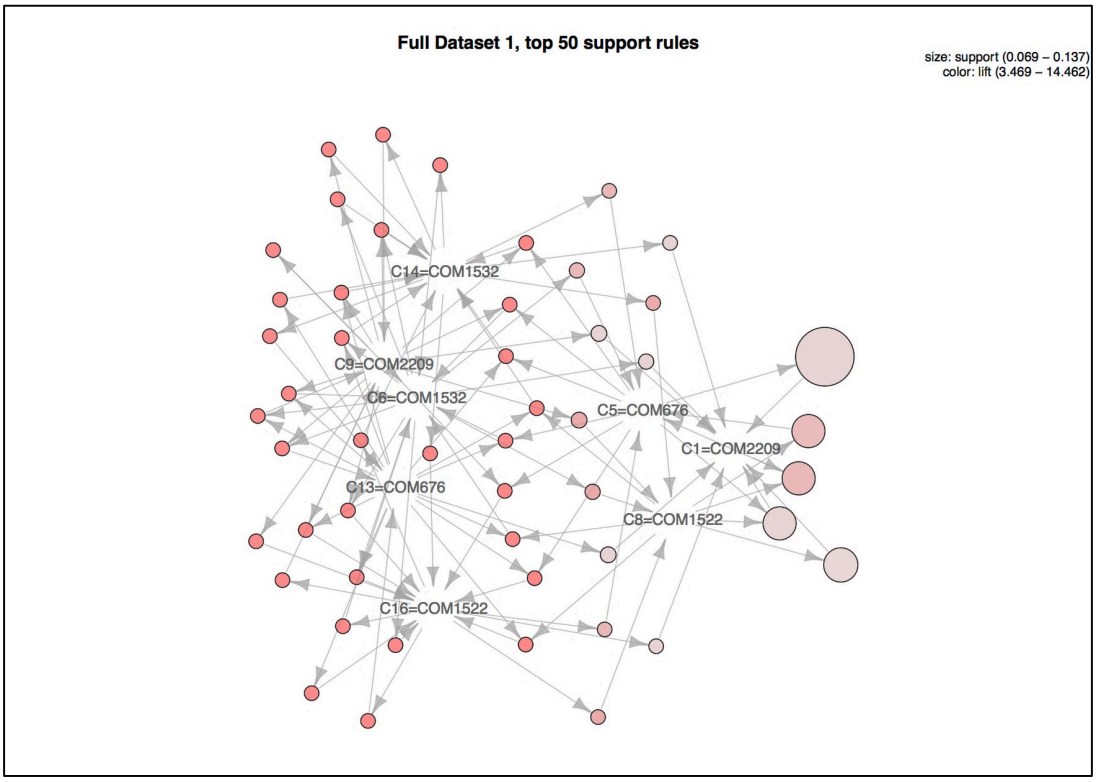


Figure 5.0.41, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from FD1.

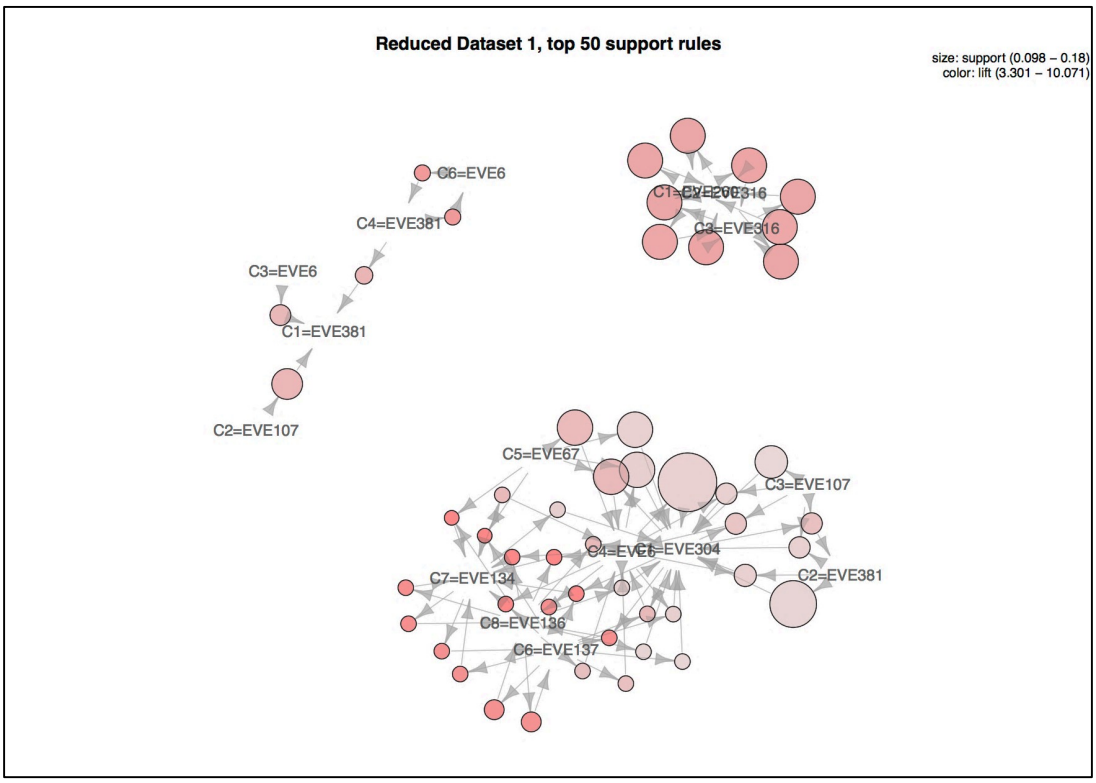


Figure 5.0.42, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from RD1.

The efficiency of the Eclat algorithm to process RD1 and FD1 is assessed by iterating the Eclat process 100 and 1,000 times (responding to *H6*). Figure 5.43 illustrates the processing time in seconds (s) for the Eclat algorithm to process RD1 and FD1 iterated 100 times. While Figure 5.44 shows the processing time in seconds (s) for the Eclat algorithm to process the RD1 and FD1 iterated 1,000 times. The black lines seen in Figure 5.43 and Figure 5.44 represent the mean processing time for the datasets after the outliers had been removed.

In contrast to Figure 5.19 that shows the efficiency of the Apriori algorithm applied to RD1 and FD1, Figure 5.43 shows the Eclat algorithm is efficient at processing FD1 compared to RD1. The greatest recorded processing time for FD1 was 0.235s while 5.342s was greatest recorded processing time for RD1. Additionally, the lowest recorded processing time for FD1 was 0.191s and 4.384s for RD1. The mean processing time of 0.207s was calculated for FD1 was calculated after a single outlier was removed. Whereas, three outliers had been removed from RD1 before a mean processing time of 4.946s was calculated. The increased processing time can be attributed to the additional item sets extracted, rules formulated along with the duplicated rules removed from RD1. Similar observations are seen in Figure 5.44.

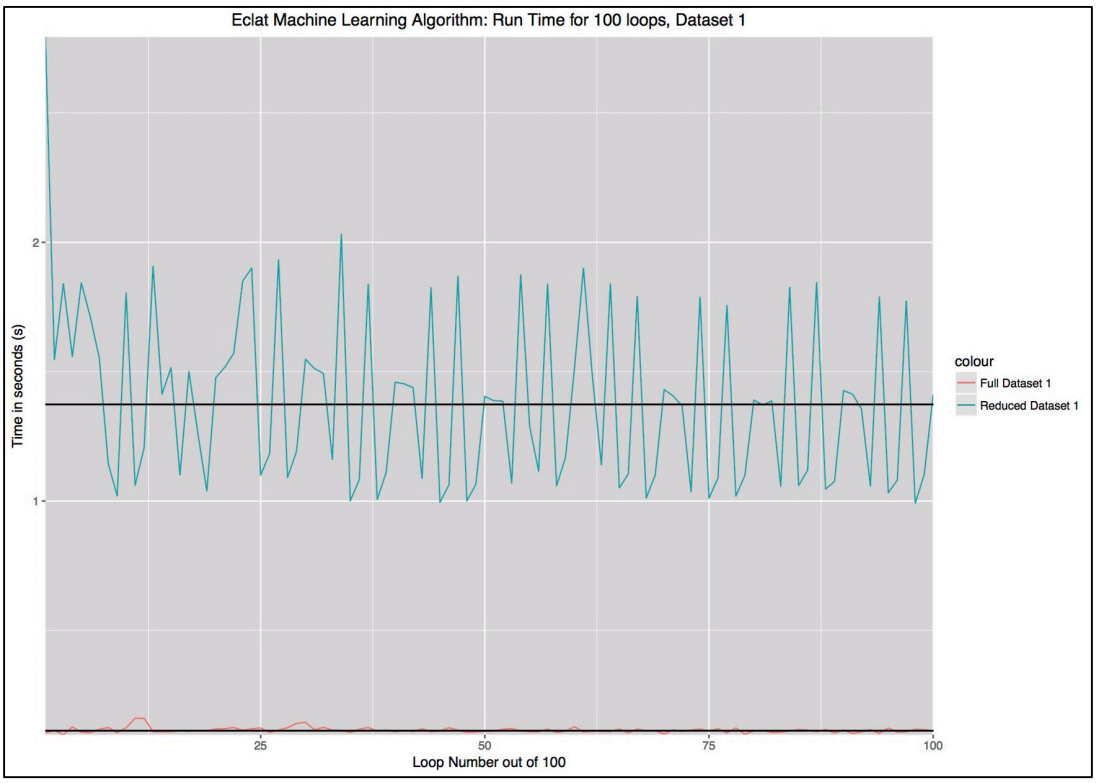


Figure 5.0.43, Illustrates the processing time in seconds(s) of the Eclat algorithm process RD1 and FD1. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

Figure 5.44 illustrates, the processing time in seconds(s) for the Eclat algorithm to process RD1 and FD1 iterated 1,000 times. As seen in Figure 5.44, the Eclat algorithm is efficient at processing FD1 compared to RD1. The greatest recorded processing time for FD1 was 0.437s while RD1 recorded 5.934s as the greatest recorded processing time. The lowest recorded processing time for FD1 was 0.185s and 4.279s for RD1. A total of 36 outliers had been removed from FD1, resulting in a mean processing time of 0.205s. In addition, a total of 57 outliers had been removed from RD1, resulting in a mean processing time of 4.921s. Inferring, the Eclat algorithm is efficient at processing FD1 compared to RD1. The aggregated results are presented in Section 5.3 along with the corresponding hypotheses tested. In the following section the results from applying the Eclat algorithm to RD2 and FD2.

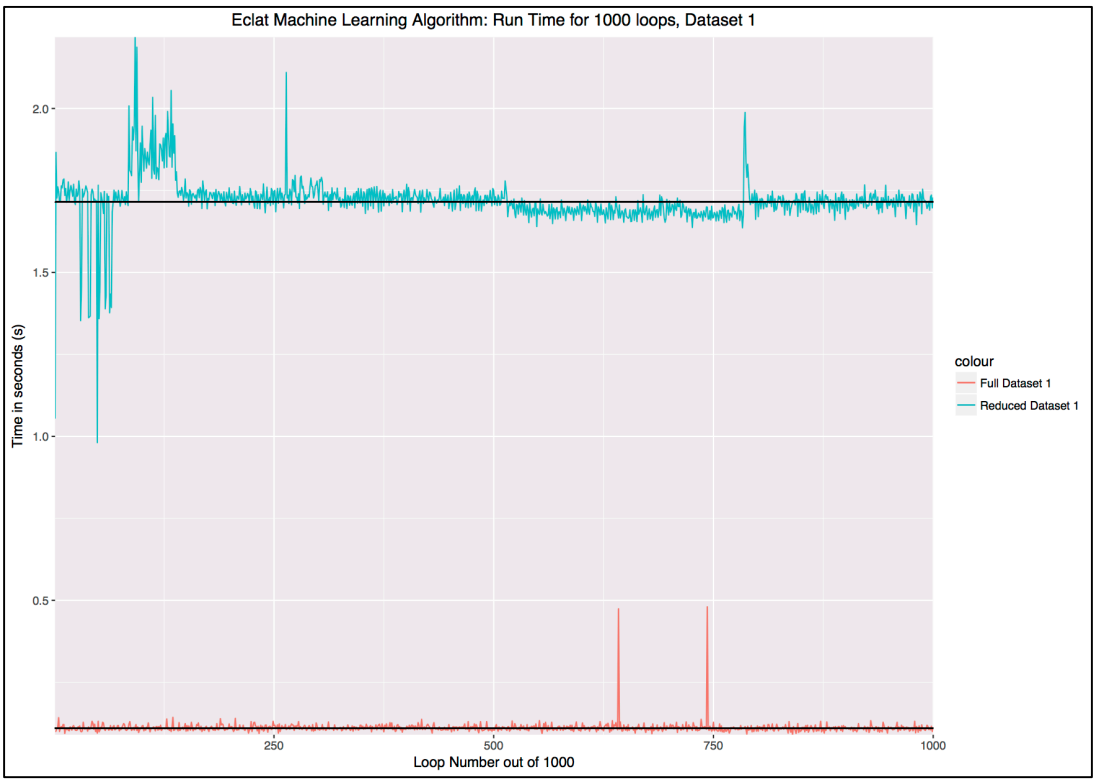


Figure 5.0.44, Illustrates the processing time in seconds(s) of the Eclat algorithm process RD1 and FD1. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

5.2.2.2 Eclat algorithm applied to Datasets Two

In this section, the results from the application of the Eclat algorithm to the RD2 and FD2 is presented below. The preliminary analysis of datasets two showed evidence of a script being utilised to interact with the Cowrie SSH honeypot. Although there was a reduction of 36.45% in the number of unique commands between RD2 and FD2. The initial assessment for datasets two suggests the minimum support value should be set to 0.9 as stated in Table 5.7. Unlike the Apriori algorithm, where the rules extracted from a dataset before removing the duplicate rules from the rule set. The Eclat algorithm extracts item sets, these are sets of frequent items within the given dataset before the rule sets are formulated then the duplicate rules are removed.

Table 5.16 presents the outcomes of the application of the Eclat algorithm to RD2 and FD2 (responding to *H2*). A total of 58,650 item sets had been extracted from the RD2 and FD2, formulating 445,168 rules. A total of 240 rules remained after the duplicate rules had been removed. The number of rules formulated and duplicated rules removed by the application of the Eclat algorithm are the same as the total number of rules extracted by the Apriori algorithm as presented in Table 5.10. Figure 5.45 and Figure 5.46 represent the total number of formulated rules, along with the support, confidence and the associated lift values.

Table 5.16, Shows the total number of rules extracted, the number of unique rules after removing the duplicate rules and the percentage of duplicate rules extracted by the Eclat algorithm applied to RD2 and FD2.

Eclat algorithm	***FD2***	***RD2***
Total number of item sets extracted	58,650	58,650
Number rules formulated	445,168	445,168
Number of unique rules after removing the duplicated rules	240	240

Figure 5.45 and Figure 5.46 show the 240 unique rules extracted from RD2 and FD2. There are three clusters of rules. One cluster of rules have a greater support and associated confidence value. While another cluster has a relatively low support and confidence values compared to the other rule clusters seen in Figure 5.45 and Figure 5.46. The remaining cluster has a relatively low support value but a greater confidence value. The mean support values are the same for both Figure 5.45 and Figure 5.46 recorded at 0.936 as well as the mean confidence values recorded at 1. There is a low lift value for the clustered rules ranging from 0 to 1.068. The lift values can range from 0 to infinity, a low lift indicates the commands in the chain are dependent on one another. The mean lift values for both Figures are 1.068. These observations are the same as the observations made in Figure 5.21 and Figure 5.22 that show the application of the Apriori algorithm on RD2 and FD2.

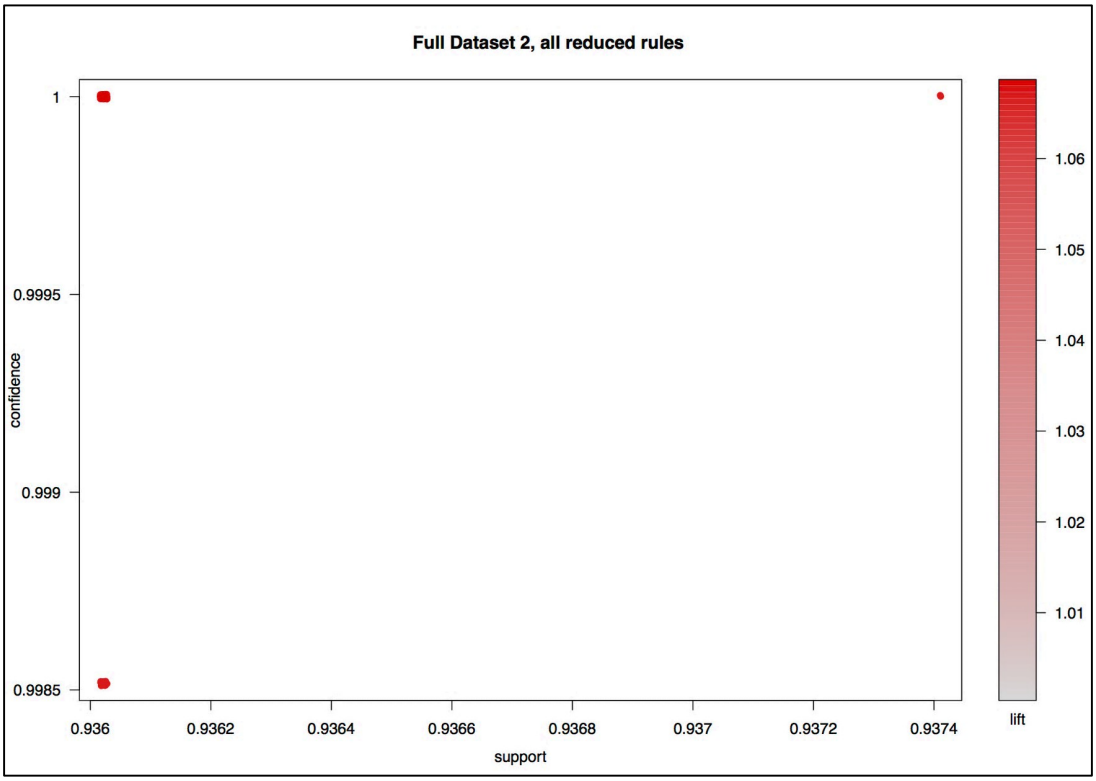


Figure 5.0.45, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Eclat algorithm from FD2.

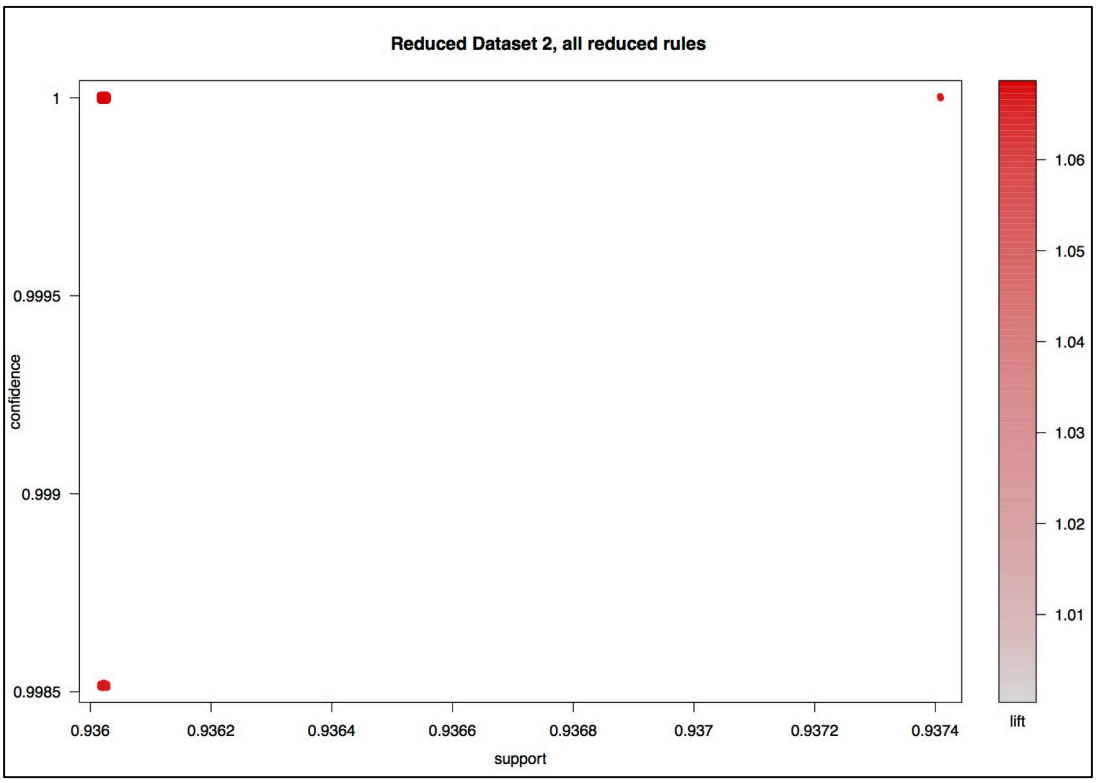


Figure 5.0.46, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Eclat algorithm from RD2.

The precision of the mined rules are based on the top 50 support based rules, since the support value is the primary parameter used to extract the item sets utilised by the Eclat algorithm (responding to *H4*). Additionally, the support parameter was chosen as the value indicates the applicability of the rule in the given dataset, as well as the support value being used to extract the top 50 rules with the Apriori algorithm. Figure 5.47 shows the top 50 support based rules mined using the Eclat algorithm on FD1. Figure 5.48 illustrating the top 50 support based rules mining for using the Eclat algorithm on RD2. Both Figure 5.47 and Figure 5.48 are similar to the associated Figure 5.45 and Figure 5.46. displaying three clusters of rules. The distribution of the top 50 support based rules for RD2 and FD2 are presented in Table 5.17.

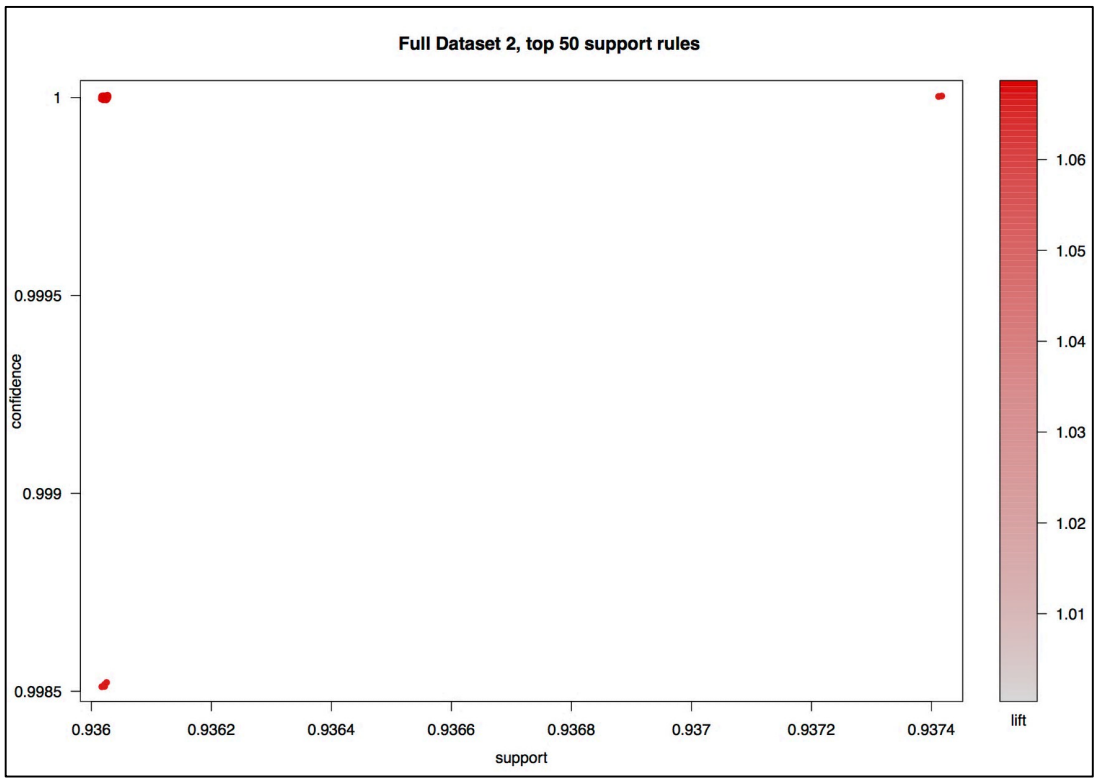


Figure 5.0.47, Illustrates a scatter plot of support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from FD2.

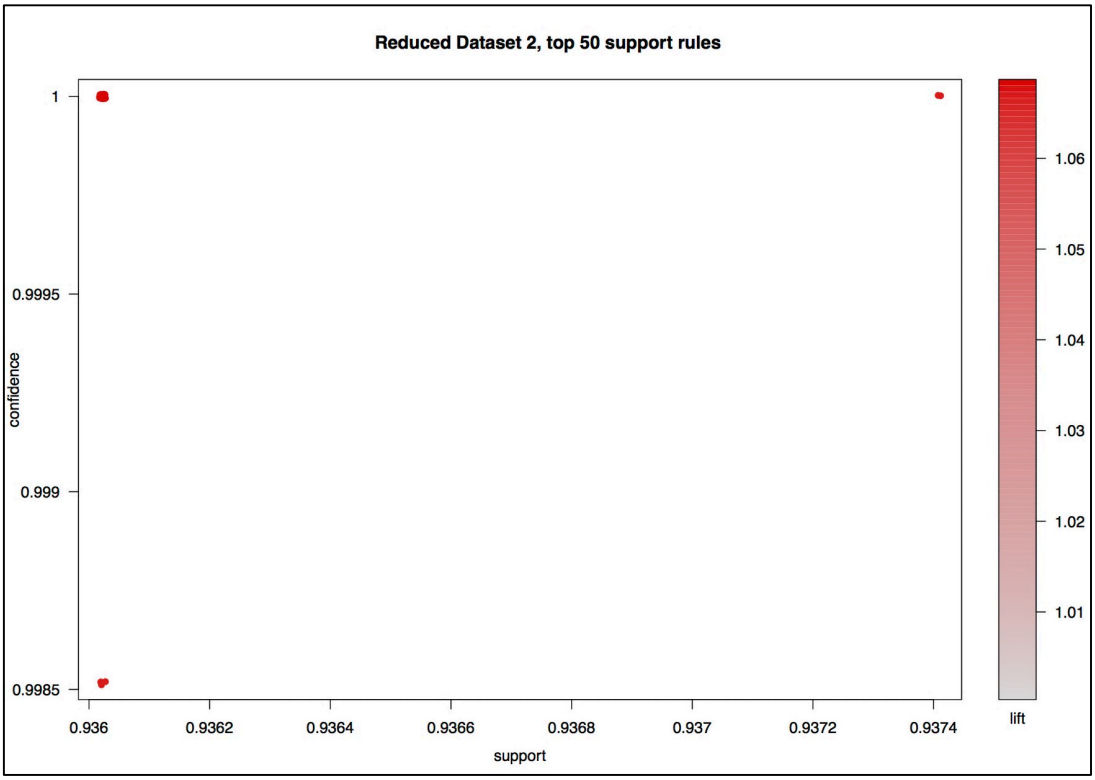


Figure 5.0.48, Illustrates a scatter plot of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from RD2.

The similarities continue between RD2 and FD2 as seen in Table 5.17. The mean support value for both RD2 and FD2 is 0.936. Indicating the extracted top 50 support based rules are applicable to RD2 and FD2. There is a low distribution of the support value of the rules, ranging between 0.936 and 0.937. The mean confidence values remain at 1, as seen in Figure 5.47 and Figure 5.48. The distribution of the confidence values for both datasets are 0.001 between the minimum and maximum. The confidence value represents the probability of the commands within a rule set appearing together. While the mean lift values of 1.068 represent the dependency of the commands in the extracted rules. The outcomes seen in Table 5.17 are similar to those seen in Table 5.11. Figure 5.49 and Figure 5.50 are the visual representation of the top 50 support based rules extracted by the Eclat algorithm from RD2 and FD2.

Table 5.17, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Eclat algorithm applied to RD2 and FD2.

Eclat algorithm applied to the second datasets	***Support***		***Confidence***		***Lift***	
	Full	**Reduced**	**Full**	**Reduced**	**Full**	**Reduced**
Minimum	0.936	0.936	0.999	0.999	1.067	1.067
First quartile	0.936	0.936	1.000	1.000	1.068	1.068
Medium	0.936	0.936	1.000	1.000	1.068	1.068
Mean	0.936	0.936	1.000	1.000	1.068	1.068
Third quartile	0.936	0.936	1.000	1.000	1.068	1.068
Maximum	0.937	0.937	1.000	1.000	1.068	1.068

A visual representation of the top 50 support based rules is presented in Figure 5.49 for FD2 and Figure 5.50 for RD2. Both figures depict the same rule set but are represented in an in different orientations. The arrows connecting the nodes and commands represent the confidence of the rule sets, with the size of the node denoting the support for each rule. The greater the support value on the scale between 0.936 and 0.937 the larger the node. While the lift values of the rules are shown by the shade of the nodes, the darker the node the greater the lift value on a scale of 1.067 to 1.068. As seen in Figure 5.49 and Figure 5.50 there are two nodes that have a greater support value of 0.937, and a lower lift value of the rules at 1.067. There are 14 key interconnected commands in 16 different placement sequence forming the top 50 support based rules. Figures 5.49 and Figure 5.50 are equivalent to the Figures 5.25 and Figure 5.26 from the application of the Apriori algorithm to RD2 and FD2.

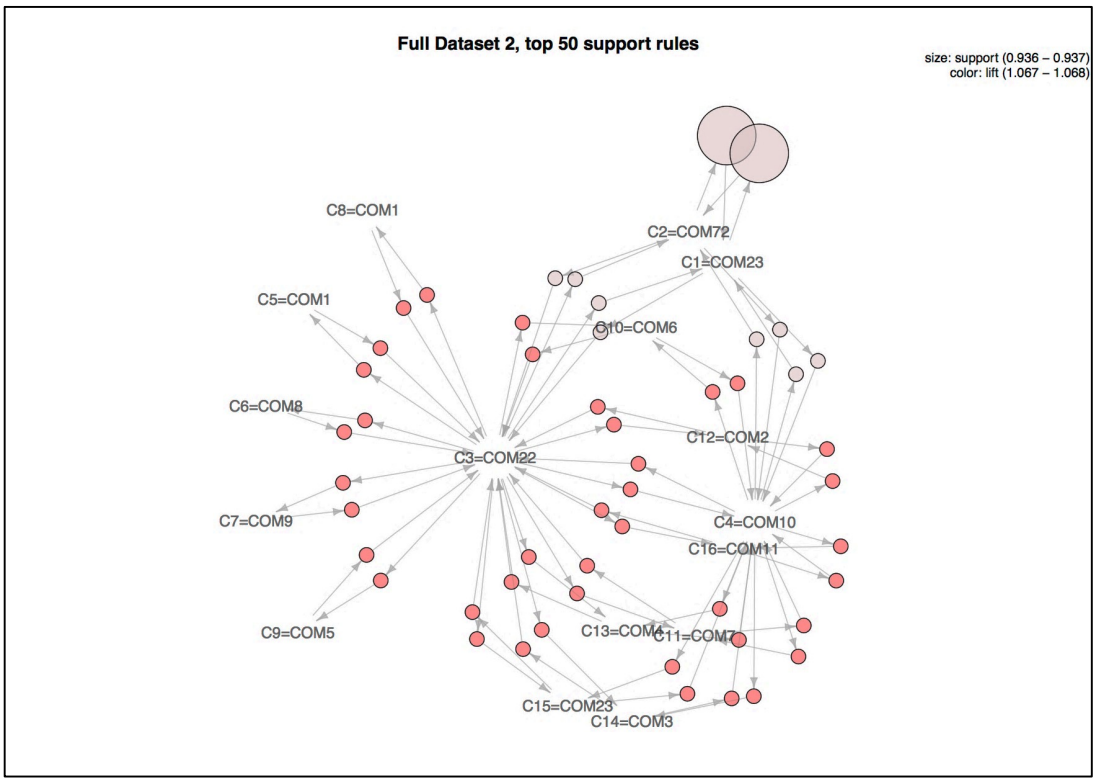


Figure 5.0.49, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from FD2.

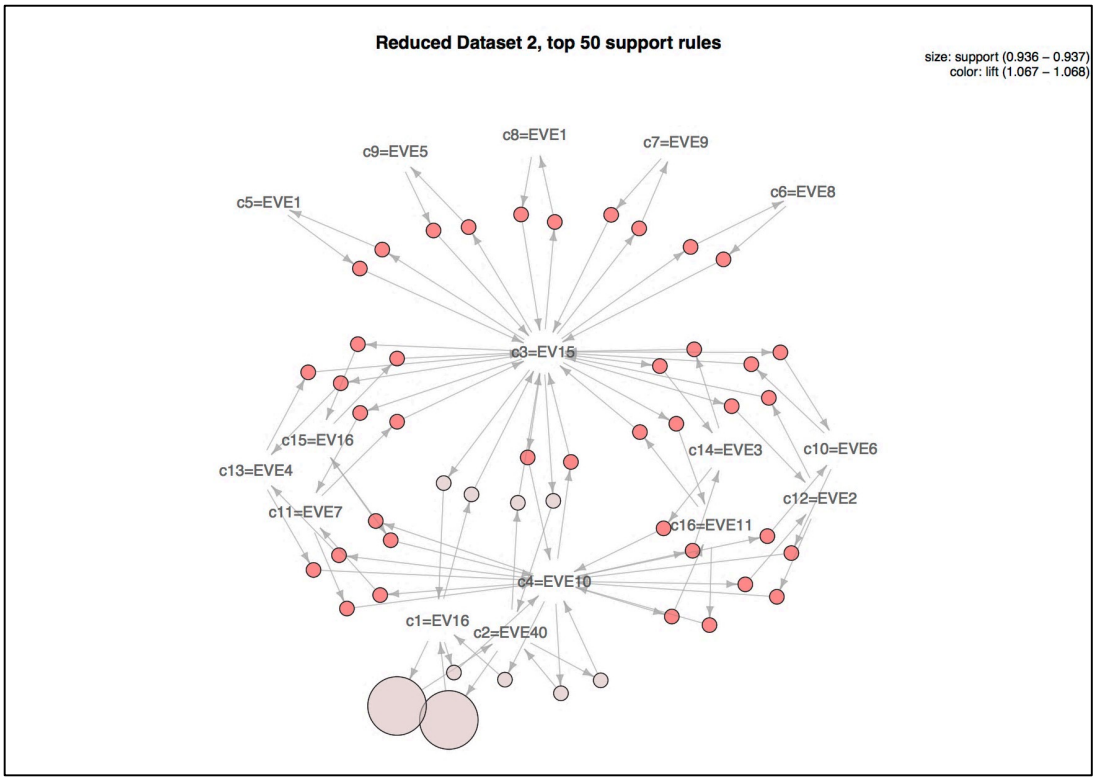


Figure 5.0.50, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from RD2.

The efficiency of the Eclat algorithm to process RD2 and FD2 are presented in Figure 5.51 and Figure 5.52. Figure 5.51 shows the processing time in seconds(s) for RD2 and FD2 iterated 100 times (responding to *H6*). The greatest processing time for FD2 was 4.269s while 4.315s was the greatest processing time for RD2. The lowest processing time for FD2 was 3.069s and the lowest processing time for RD2 was 3.043s. The black lines represent the mean values for both datasets after the outliers had been removed. No outliers had been removed for both datasets. The mean processing time in seconds(s) for FD2 was 3.464s and 3.692s was the mean processing time for RD2.

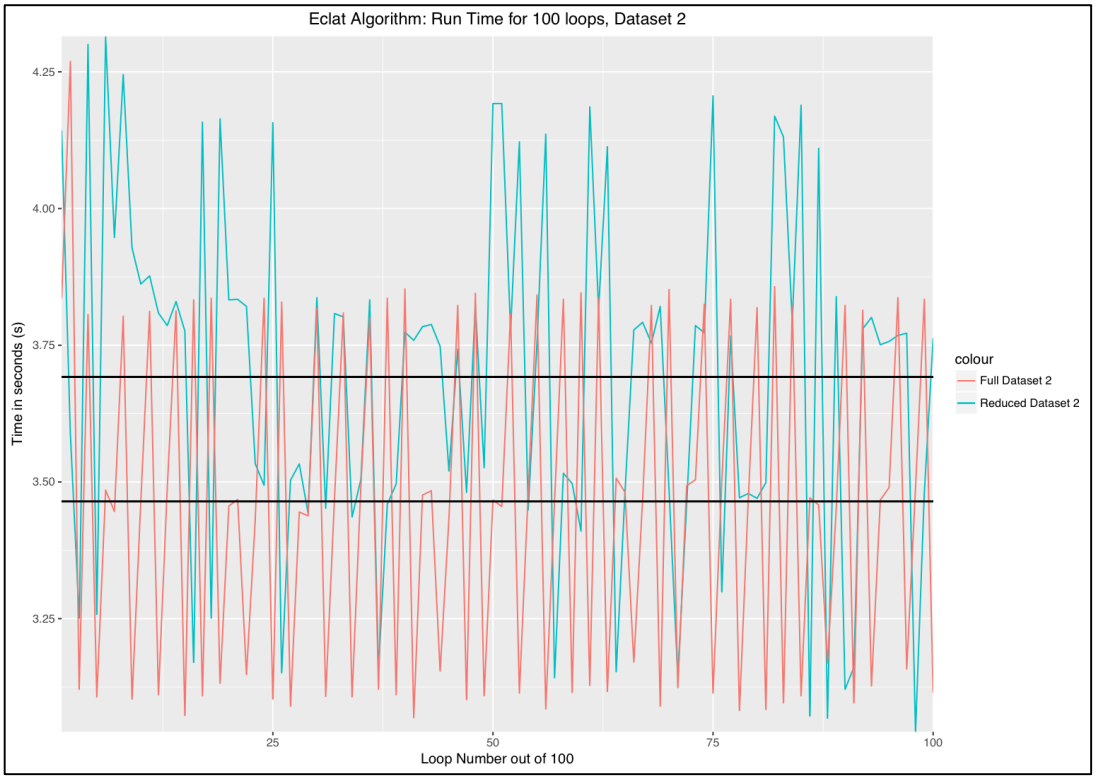


Figure 5.0.51, Illustrates the processing time in seconds(s) of the Eclat algorithm process RD2 and FD2. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

Figure 5.52, shows the efficiency of the Eclat algorithm processing RD2 and FD2 iterated 1,000 times (responding to *H6*). The greatest recorded processing time was 4.896s for FD2. The greatest reduced processing time for RD2 recorded 5.284s. The lowest recorded processing time for FD2 was 3.053s and 3.017s for RD2. The data points for both datasets overlap, however predominantly the data points for FD2 are lower than 3.5s, while the data points for RD2 are greater than 4.0s. The mean processing time for FD2 in Figure 5.52 was 3.53s after one outlier was removed. While the mean processing time for RD2 was 3.635s after four outliers had been removed. The aggregated results are presented in Section 5.3 along with the corresponding hypotheses tested. In the following section the results from applying the Eclat algorithm to RD3 and FD3.

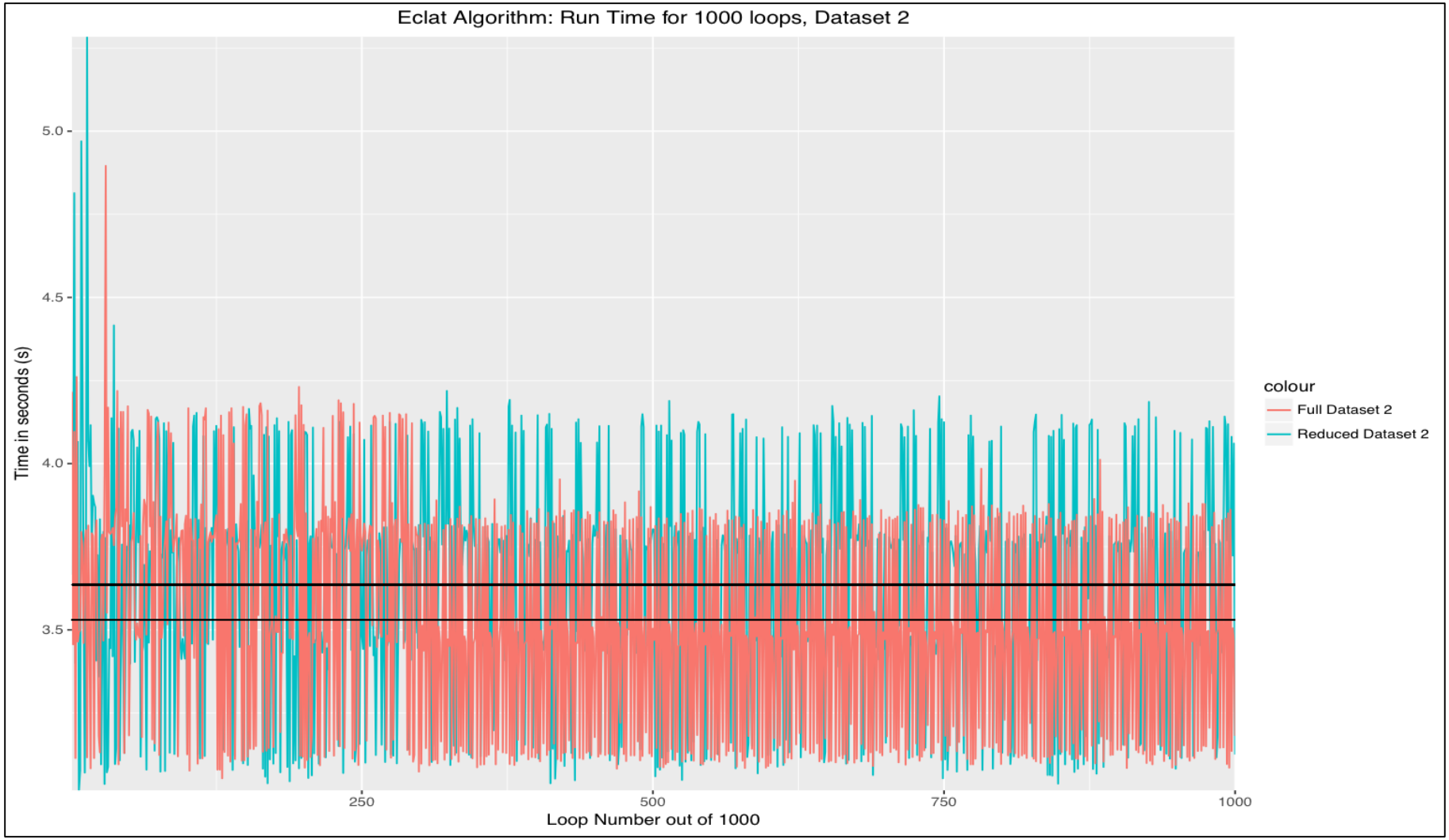


Figure 5.0.52, Illustrates the processing time in seconds(s) of the Eclat algorithm process RD2 and FD2. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

5.2.2.3 Eclat algorithm applied to Datasets Three

In this section, the results from the application of the Eclat algorithm on RD3 and FD3 are be presented. The pre-processing phase in Section 4.2.3 of dataset three showed FD3 had a total of 748 unique commands within a dataset of 318 sessions. While the clustering process resulted in RD3 containing a total of 338 unique commands within a dataset of 318 sessions. Table 5.18 presents the extracted item sets, the number of rules formulated and the number of unique rules after duplicated rules have been removed for t RD3 and FD3 (responding to *H2*). The support parameter for the Eclat algorithm was set at 0.01, as stated in Table 5.7.

FD3 had 239 item sets extracted, while 285 item sets had been extracted from RD3, as seen in Table 5.18. Item sets are extracted are frequent command sets occurring within the given dataset. The rules are formulated based on the extracted item sets before the duplicate rules are removed. A total of 460 rules had been formulated before 400 duplicated rules had been removed from FD3 compared to 125 unique rules mined by the Apriori algorithm. The number of item sets extracted from the RD3 was greater than FD3 at 285. But 459 rules had been formulated from RD3 before 397 duplicated rules had been removed, leaving 62 unique rules as mentioned in Table 5.18. Compared to 170 unique rules mined by the Apriori algorithm. The support, confidence and lift values of the unique rules extracted by the Eclat algorithm applied to FD3 are presented in Figure 5.53 and for RD3 are presented in Figure 5.54.

Table 5.0.18, Shows the total number of rules extracted, number of unique rules after removing the duplicate rules and the percentage of duplicate rules extracted by the Eclat algorithm applied to RD3 and FD3.

Eclat algorithm	***FD3***	***RD3***
Total number of item sets extracted	239	285
Number rules formulated	460	459
Number of unique rules after removing the duplicated rules	60	62

Figure 5.53 shows the 60 unique rules mined by the Eclat algorithm on FD3, while Figure 5.54 shows the 62 unique rules extracted by the Eclat algorithm from RD3. Both Figures are similar, with the support values of the mined rules between 0.01 and 0.1. Though, as seen in Figure 5.53, much of the rules depicted have a support value are between 0.01 and 0.02, it is reflected in the mean support value of 0.015. The mean support value for RD3 depicted in Figure 5.54 was also 0.015. However, the associated confidence values for the mined rule sets are between 0.8 and 1, the mean confidence value for the rule set depicted in Figure 5.53 was 0.971. Although the mean confidence value of Figure 5.54 was 0.933. The lift values for the rules range between 0 and 80. A lower lift value indicates the dependency of the commands within an extracted rule. The mean lift value for FD3 shown in Figure 5.53 was 58.459 compared to the mean lift value for RD3 depicted in Figure 5.54 at 43.602. In the forthcoming section, the top 50 support based rules are extracted.

As presented in Table 5.18, a total of 60 unique rules had been extracted by the Eclat algorithm from FD3. Consequently, the top 50 unique support based rules are depicted in Figure 5.55 are similar to Figure 5.53. Furthermore, there had been 62 unique rules extracted from RD3 by the Eclat algorithm. Figure 5.56 of the top 50 unique support based rules for RD3 is similar to Figure 5.54 depicting the extracted 62 unique rules as seen in Table 5.18. The distribution of the rule sets depicted in Figure 5.55 of FD3 and Figure 5.56 of RD3 are presented in Table 5.19. The distribution of the rule sets is used to evaluate the precision of the patterns in the extracted rule sets by the Eclat algorithm from the RD3 and FD3.

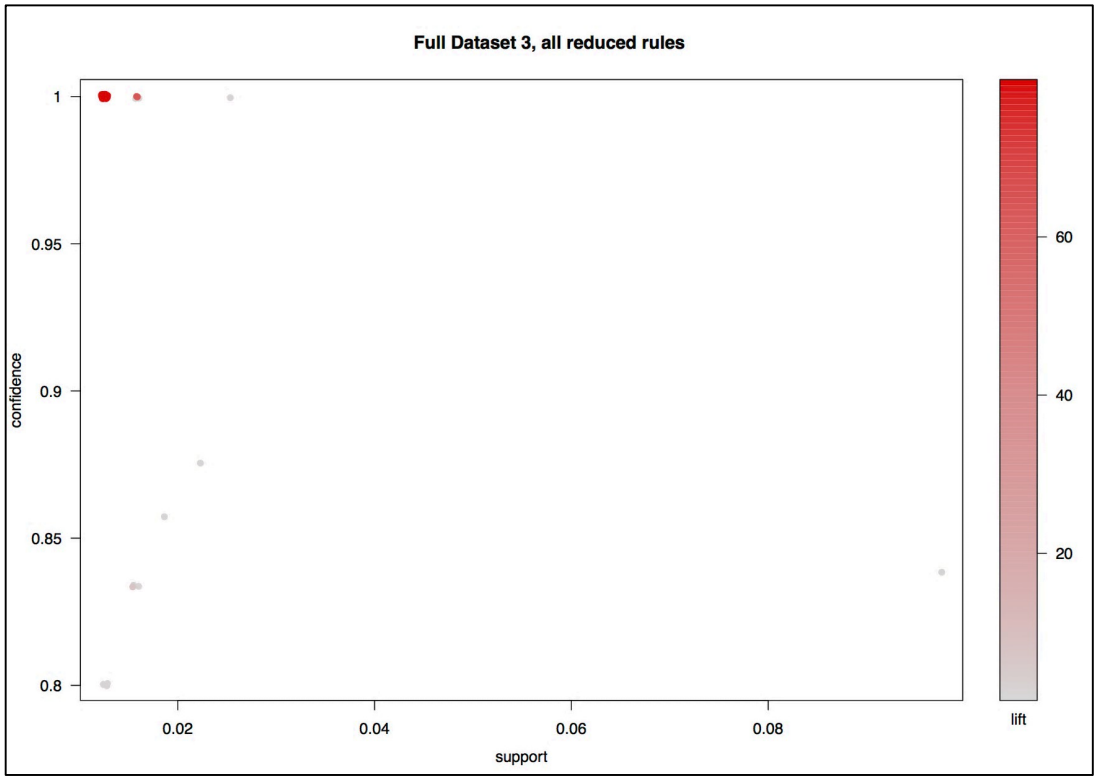


Figure 5.0.53, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Eclat algorithm from FD3.

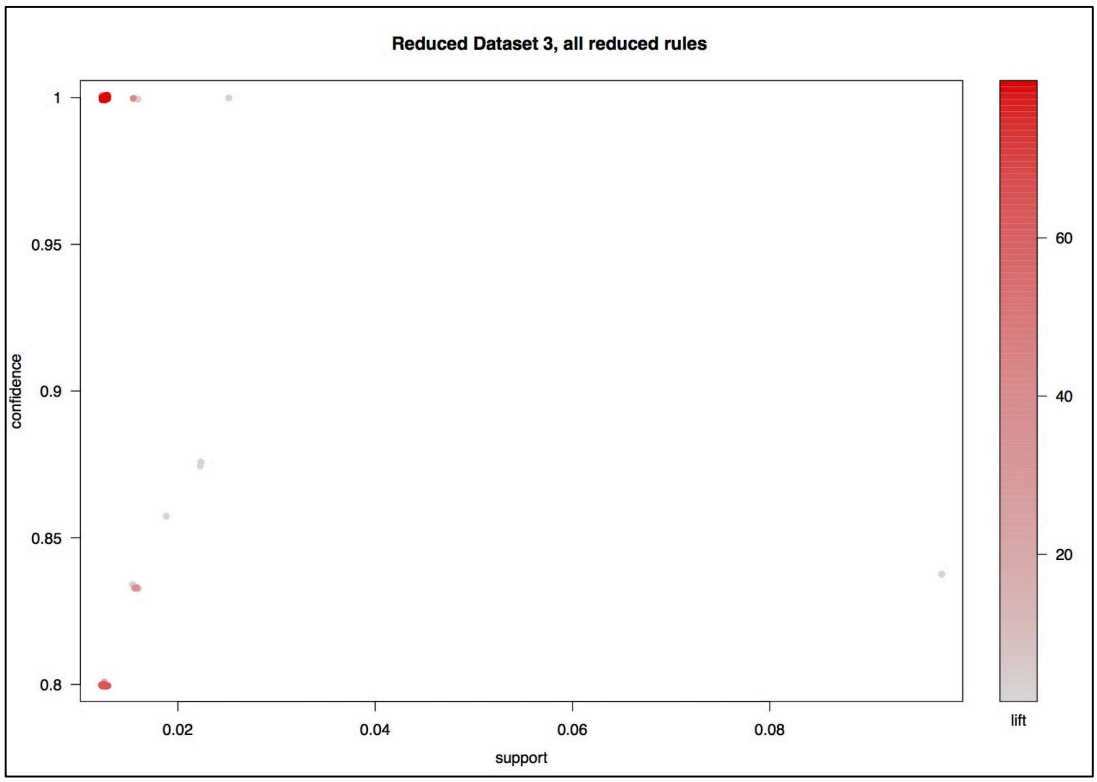


Figure 5.0.54, Illustrates a scatter plot of support, confidence and lift values of the total rule set extracted by the Eclat algorithm from RD3.

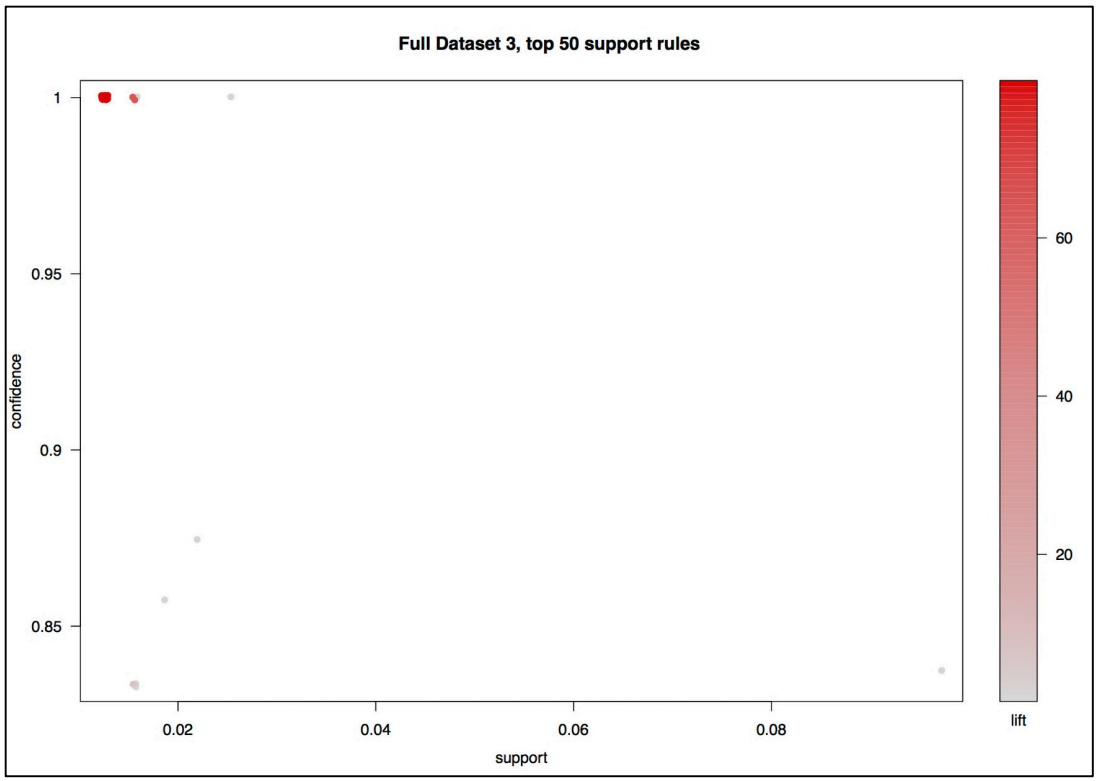


Figure 5.0.55, Illustrates a scatter plot of support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from FD3.

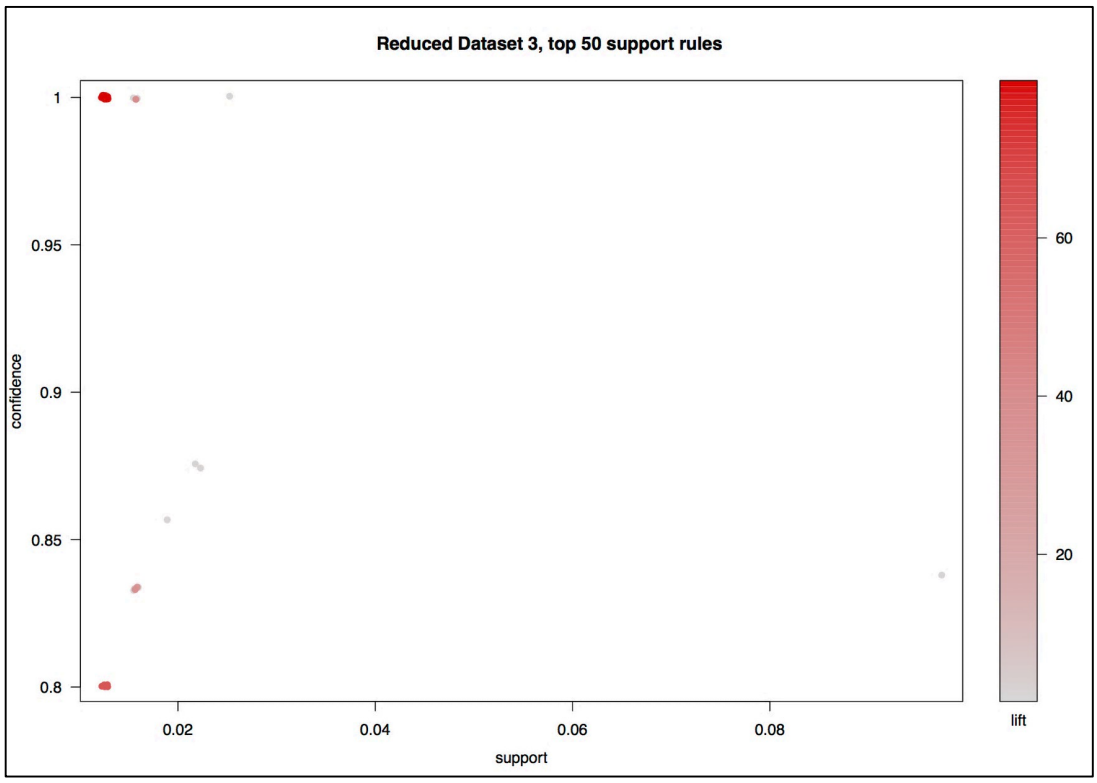


Figure 5.0.56, Illustrates a scatter plot of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from RD3.

Table 5.19 presents the top 50 unique support based rules extracted by the Eclat algorithm applied to RD3 and FD3 (responding to *H4*). As seen in Table 5.19, the distribution of the support value for the rule sets is the same for RD3 and FD3, with the mean support value recorded at 0.015. The mean support value in Table 5.19 was lower than the mean support value of 0.022 for FD3 and 0.023 for RD3 as presented in Table 5.13 for the Apriori algorithm. The distribution of the confidence values of the top 50 support based rules is between the minimum of 0.8 and a maximum of 1, for RD3. The support value indicates the applicability of the mined rules to the given datasets. However, the confidence value distribution of FD3 rules presented in Table 5.19 ranged between 0.833 and 1. Furthermore, the mean confidence values are greater for FD3 at 0.981 compared to 0.949 for RD3, this was also observed in Figure 5.53 and Figure 5.54 depicting the total extracted unique rule sets. The confidence of a rule indicates the probability of a command occurring based on the presence of a given command in the sequence. Yet the mean lift value for RD3 was 50.870 compared 63.530 for FD3. A visual representation of the top 50 unique support based rules extracted by the Eclat algorithm applied to RD3 and FD3 are presented in Figure 5.57 and Figure 5.58 respectively.

Table 5.19, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Eclat algorithm applied to RD3 and FD3.

Eclat algorithm applied to the third datasets	***Support***		***Confidence***		***Lift***	
	Full	**Reduced**	**Full**	**Reduced**	**Full**	**Reduced**
Minimum	0.013	0.013	0.833	0.800	2.120	2.120
1st quartile	0.013	0.013	1.000	0.862	79.500	34.310
Medium	0.013	0.013	1.000	1.000	79.500	58.300
Mean	0.015	0.015	0.981	0.949	63.530	50.870
3rd quartile	0.013	0.013	1.000	1.000	79.500	79.500
Maximum	0.097	0.097	1.000	1.000	79.500	79.500

Figure 5.57, is a visual depiction of the top 50 unique support based rules extracted by the Eclat algorithm applied to the full dataset. The confidence values of the rules are represented by the arrows connecting the commands and nodes together. While the size of the nodes represents the support value of the rules, ranging between 0.013 and 0.097. As seen in Figure 5.57 there is a single large node denoting the maximum support of 0.098 seen in Table 5.19. The lift values of the rules are represented through the colour of the nodes, nodes with a greater lift are illustrated in a darker shade. From Figure 5.57, each rule cluster has a varying associated lift value. The lift values range from 2.12 and 79.5. The largest cluster of rules has the greater associated lift value. Figure 5.58 visually depicts the top 50 unique support based rules extracted by the Eclat algorithm applied to the RD3. As seen in Figure 5.57, there is a single larger node representing the maximum support value of 0.098 seen in Table 5.19. There are four clusters of rules that form the top 50 rule set. Unlike Figure 5.57 the largest cluster of rules in Figure 5.58 has nodes with varying lift values, between 2.12 and 79.5. There is a total of 16 interconnected commands that form the top 50 rule set depicted in Figure 5.57, in 20 different sequence placements. Yet depicted in Figure 5.58, there are a total of 17 interconnected commands in 22 varying placements in a sequence.

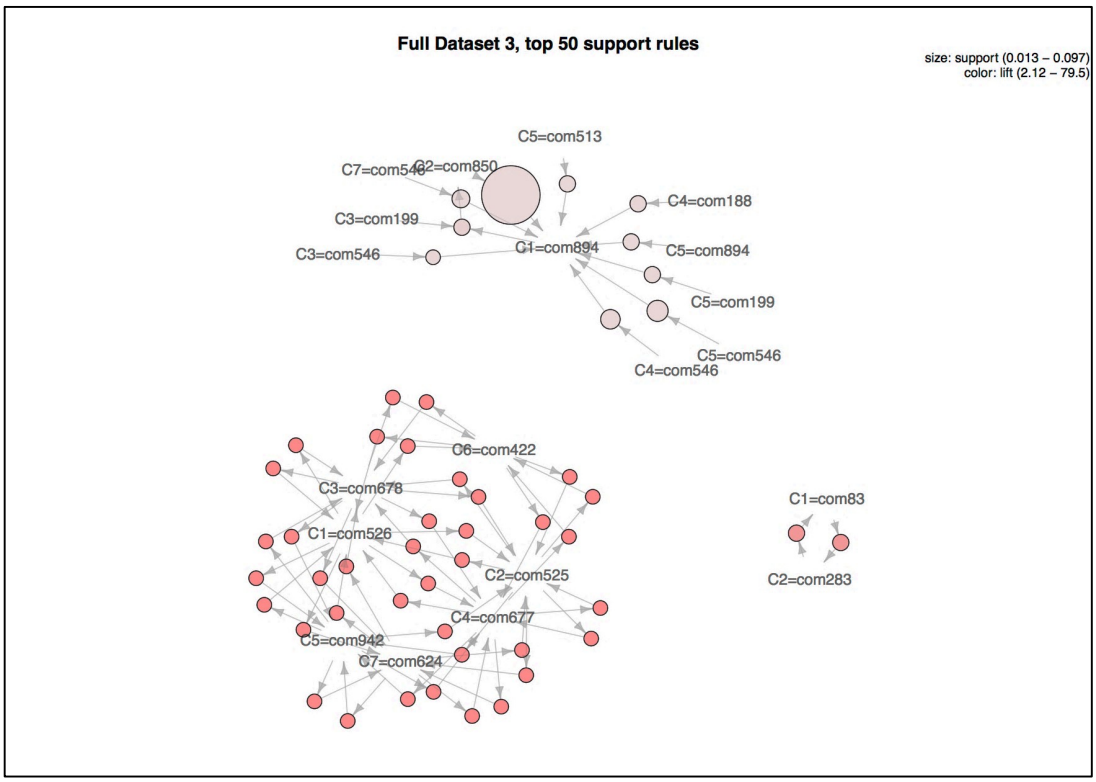


Figure 5.0.57, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from FD3.

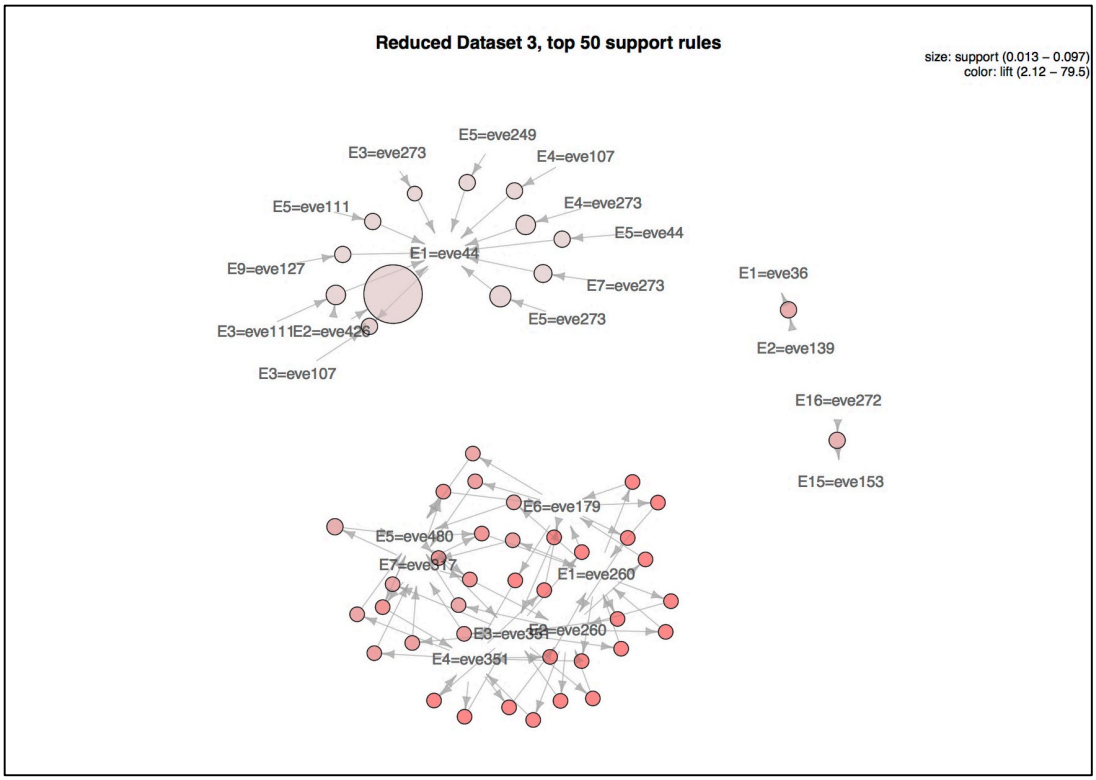


Figure 5.0.58, is a visual representation of the support, confidence and lift values of the top 50 support based rule set extracted by the Eclat algorithm from RD3.

Figure 5.59 displays the efficiency of the Eclat algorithm to process RD3 and FD3. The process was iterated 100 times, with the time in seconds(s) for the Eclat algorithm to process the datasets (responding to *H6*). The greatest recorded time was 0.444s for FD3 and 0.290s for RD3. The lowest recorded processing time for the full dataset was 0.175s with RD3 recording 0.207s. After removing three outliers from both datasets, the mean value for both datasets had been calculated and are illustrated by the black lines. The mean processing time for FD3 was 0.209s while the mean processing time for RD3 was 0.230s.

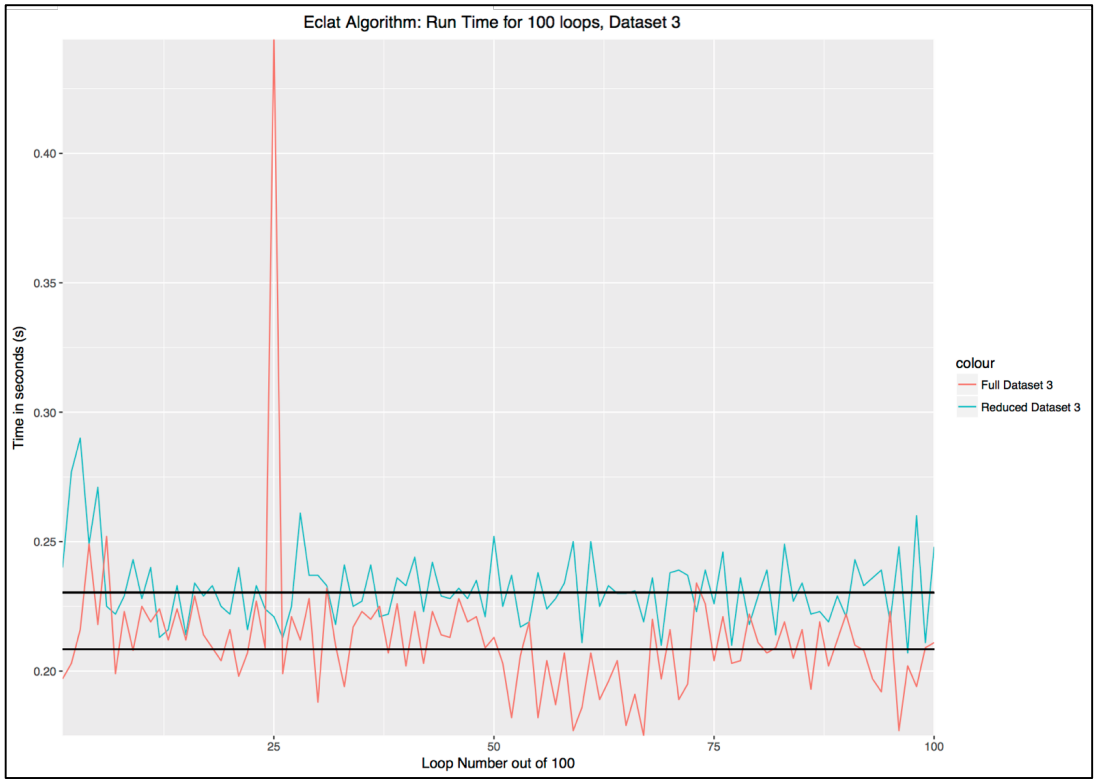


Figure 5.0.59, Illustrates the processing time in seconds(s) of the Eclat algorithm process RD3 and FD3. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

Figure 5.60 shows the processing time of the Eclat algorithm applied to RD3 and FD3 iterated 1,000 times (responding to *H6*). The greatest recorded processing time was 0.449s for FD3 and 0.473s for RD3. The lowest processing time for the full dataset was 0.174s with RD3 recorded at 0.205s. After removing 13 outliers from FD3 and 30 from RD3, the mean for both datasets had been calculated and are illustrated by the black lines. The mean processing time for FD3 was 0.206s while the mean processing time for RD3 was 0.231s. The aggregated results are presented in Section 5.3 along with the corresponding hypotheses tested

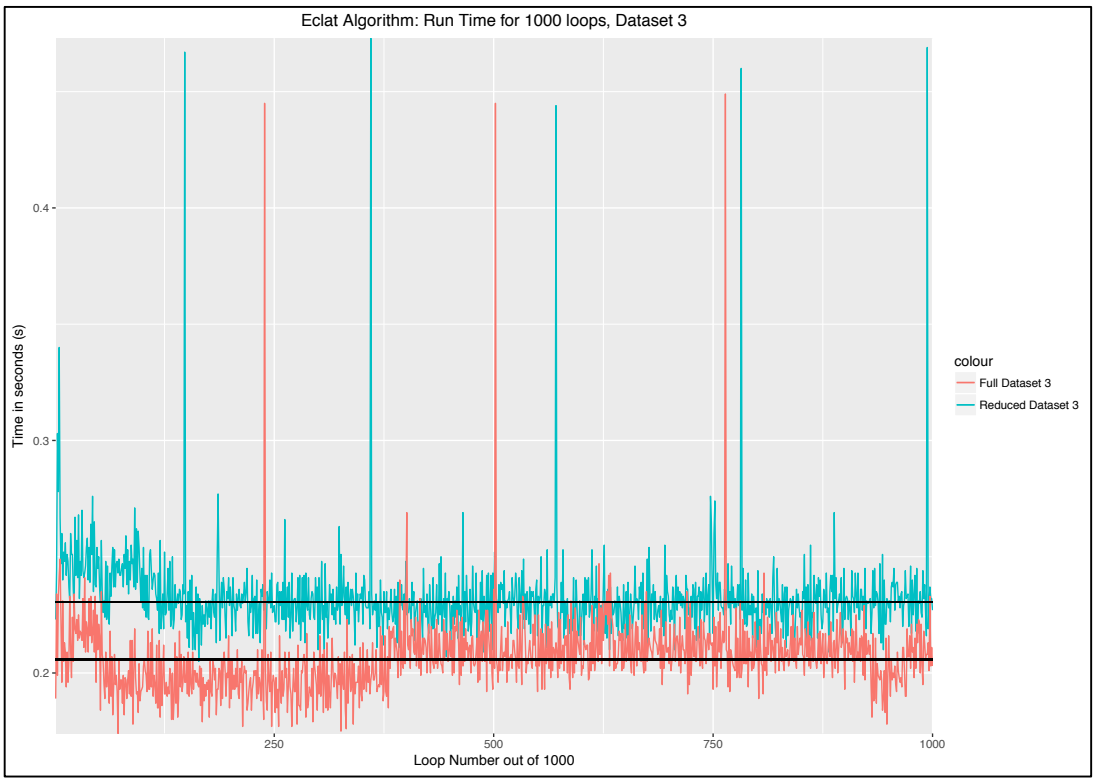


Figure 5.0.60, Illustrates the processing time in seconds(s) of the Eclat algorithm process RD3 and FD3. The process was iterated 1,000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed.

5.3 Aggregation of Results

Upon completing the experiments, the results are aggregated according to the hypotheses they answer. Analysis of the results is conducted within the Discussion Chapter 6. The chosen machine learning algorithms are the Naïve Bayes, Markov chain, Apriori and the Eclat algorithms. The results from applying machine learning algorithms to the reduced datasets and their respective full datasets are presented in this section.

The probabilistic classification algorithms chosen to be applied to the datasets are the Naïve Bayes and Markov chain algorithms. The association rule mining algorithms chosen are the Apriori and Eclat algorithms. The assessments had been based on whether additional patterns can be extracted from the reduced datasets and their respective full datasets. As well as evaluating whether the extracted patterns from the reduced datasets are more precise compared to those patterns extracted from their relevant full datasets. Along with assessing the efficiency of the algorithms to extract the patterns from the reduced datasets and their respective full datasets. The hypotheses and associated research questions are evaluated in the subsequent Discussion Chapter 6. The experimental results to test the six hypotheses of this study as outlined in Section 3.2 are elaborated below.

5.3.1 Hypothesis One

H1: More class patterns can be extracted from the reduced sequence of commands datasets, compared to those extracted from their respective full datasets by the selected probabilistic classification algorithms.

Hypothesis one (*H1),* states more class patterns can be extracted from the reduced datasets by the selected probabilistic classification algorithms compared to the number of class patterns extracted from their respective full datasets. To assess whether additional patterns can be extracted from the reduced datasets compared to their respective full datasets, the True Positive (TP) rate was compared. Since the Naïve Bayes algorithm identifies patterns within a given dataset based on the selected feature. Inferring true patterns are seen if the classification algorithm can predict instances within a given dataset, irrespective of whether the instances are correctly classified.

Table 5.20, shows the TP rates obtained from applying the Naïve Bayes algorithms to the reduced datasets and their respective full datasets. As seen in Table 5.20, RD1 has a higher TP rate than FD1. While RD2 and RD3 have the same TP rate as their respective full datasets. Depending on the dataset presented it can be suggested additional patterns can be extracted by the Naïve Bayes algorithm from the reduced datasets compared to their respective full datasets.

Table 5.20, Shows the aggregated True Positive (TP) rate of the trained Naïve Bayes classifier applied to the test sets of the three reduced datasets and their respective full datasets, along with the difference between the rates.

Naïve Bayes	***RD1***	***FD1***	***RD2***	***FD2***	***RD3***	***FD3***
TP Rate	77.027%	71.623%	99.505%	99.505%	11.111%	11.111%

To assess whether additional patterns can be extracted from the reduced datasets compared to their respective full datasets by the Markov chain algorithm, the degree of freedom of the datasets are compared. A Markov chain algorithm determines the probability of the next state or command occurring with the probability values presented as a transition matrix. To evaluate if additional patterns can be extracted by the Markov chain algorithm the degree of freedom for the reduced datasets and their respective full datasets are compared. The degree of freedom is the number of independent values within a dataset before a subsequent value can be predicted.

Table 5.21, Shows the degree of freedom values calculated for the three reduced datasets and their respective full datasets. Table 5.20 shows for both the full and reduced datasets two and three the TP rate is the same. In contrast, the results shown in Table 5.21 all three reduced datasets had a lower degree of freedom compared to their respective full datasets, suggesting additional patterns can be

extracted by the Markov chain algorithm from the reduced datasets compared to their respective full datasets. Resulting in the acceptance of hypothesis *H1*.

Table 5.21, Shows the aggregated degree of freedom for reduced datasets and their respective full datasets calculated by the Markov chain algorithm.

Markov chain	***RD1***	***FD1***	***RD2***	***FD2***	***RD3***	***FD3***
Degree of freedom	1.279e+14	1.419e+14	667,627,624	778,688,000	3.070e+14	1.943e+15

5.3.2 Hypothesis Two

H2: More patterns can be extracted from the rule sets of the reduced sequence of commands datasets, compared to those extracted from their respective full datasets by the selected association rule mining algorithms.

Hypothesis two (*H2)* states more rule sets can be extracted from the reduced datasets by the selected association rule mining algorithms compared to the number of class patterns extracted from their respective full datasets. An association rule mining algorithm calculates the probability of a sequence of instances occurring based on the frequency they appear in the given dataset to formulate rules.

To assess whether additional patterns can be extracted by the Apriori algorithm from the reduced datasets and their respective full datasets, the number of rules remaining once the redundant rules had been removed are compared. As the extracted rules are formulated based on patterns identified between the mined frequent items within the datasets. Table 5.22, Shows the number of rules remaining once the redundant rules had been removed from the reduced datasets and their respective full datasets by the Apriori algorithm. As seen in Table 5.22, the number of remaining rules for RD1 and RD3 are greater compared to the number of remaining rules for their respective full datasets. However, the number of the number of remaining rules for RD2 and FD2 is the same, as also seen with the Naïve Bayes algorithm.

Table 5.22, Shows the aggregated unique rule sets for the reduced datasets and their respective full datasets after the duplicated rules had been removed by the Apriori algorithm

Apriori	***RD1***	***FD1***	***RD2***	***FD2***	***RD3***	***FD3***
Number of unique rules after the duplicated rules had been removed	1,893	98	240	240	170	125

To assess whether additional patterns can be extracted from the reduced datasets and their respective full datasets by the Eclat algorithm, the number of rules remaining once redundant rules had been removed is compared. As the extracted rules are formulated based on patterns identified between the frequent items within the datasets. The Eclat algorithm uses a depth-first search technique. Table 5.23 presents the number of rules remaining once the redundant rules had been removed for the reduced

datasets and their respective full datasets by the Eclat algorithm. As seen in Table 5.23, RD1 and RD3 have a greater number of rules remaining once the redundant rules had been removed compared to their respective full datasets. While RD2 and FD2 had the same number of rules remaining, as was also seen with the Naïve Bayes and Apriori algorithms. Resulting in the acceptance of hypothesis *H2*.

Table 5.23, Shows the aggregated unique rule sets for the reduced datasets and their respective full datasets after the duplicated rules had been removed by the Eclat algorithm

Eclat	***RD1***	***FD1***	***RD2***	***FD2***	***RD3***	***FD3***
Number of unique rules after the duplicated rules had been removed	1,441	64	240	240	62	60

5.3.3 Hypothesis Three

H3: The class patterns extracted by the probabilistic classification algorithms from the reduced sequence of commands datasets are more precise, compared to those extracted from their respective full datasets.

Hypothesis three (*H3)* states the class patterns extracted by the probabilistic classification algorithms from the reduced datasets are more precise compared to those extracted from their respective full datasets. The precision of the class patterns extracted by the Naïve Bayes algorithm is assessed using a confusion matrix. The precision, accuracy, sensitivity, F1 score and error rate of the classified instances are used to evaluate the overall precision of the class pattern extracted by the algorithm. As outlined in Section 5.1.1 the training feature selected to train the classifier was the same for the reduced datasets and their respective full datasets.

Table 5.24 shows the results for the precision, accuracy, sensitivity, F1 score and Error rate of the trained classifiers for each dataset.

Table 5.24 shows the precision of the Naïve Bayes classifier for the reduced datasets and their respective full datasets. The precision is used to measure the correctness of the classification algorithm when classifying data. The results show FD1 has a greater precision rate compared to RD1, with a difference of 0.014 between the values. In contracts, RD3 has a greater precision rate compared to FD3, with a difference of 0.062 between the values. While the precision rate is the same for RD2 and FD2.

The accuracy rate shown in Table 5.24 is used to measure the probability of the truly classified data. Unlike the precision rate, RD1 has a greater accuracy rate compared to FD1. However, the accuracy rate for the RD2 and RD3 are the same as their respective full datasets.

Sensitivity is also known as the recall (true positive rate), this is the number of objects correctly classified as positive. As seen in Table 5.24, FD1 has a greater sensitivity rate compared to RD1, with a difference of 0.032 between the values. Yet RD3 recorded a greater sensitivity rate compared to FD3. The sensitivity rates obtained for the RD2 and FD2 had been the same.

Table 5.24 shows the F1 score used to measure the predictive probability of the classification produced by calculating a balanced mean between the precision and sensitivity. The F1 score for FD1 is greater compared to the F1 score obtained for RD1, with a difference of 0.028 between the values. Whereas the F1 score obtained for RD3 is greater compared to FD3. While the F1 score obtained for the RD2 and FD2 are the same.

The error rate measures the rate of incorrectly classified data, a lower error rate is desirable. Table 5.24, shows RD1 has a lower error rate compared to FD1. While the error rate for RD2 and RD3 are the same as their respective full datasets.

Table 5.24, Shows the aggregated precision of the trained Naïve Bayes classifier applied to the three reduced datasets and their respective full datasets

Naïve Bayes	***RD1***	***FD1***	***RD2***	***FD2***	***RD3***	***FD3***
Precision	0.646	0.660	0.998	0.998	0.161	0.099
Accuracy	0.770	0.716	0.995	0.995	0.111	0.111
Sensitivity (Recall)	0.429	0.461	0.667	0.667	0.119	0.087
F1 Score	0.515	0.543	0.799	0.799	0.137	0.092
Error Rate	0.230	0.284	0.005	0.005	0.889	0.889

A Markov chain algorithm determines the probability of the next state or command occurring and is presented as a transition matrix. To assess whether class patterns extracted by the Markov chain algorithm are more precise from the reduced datasets compared to their respective full datasets, the mean and standard deviation values of the standard error matrices, lower endpoint matrices and upper endpoint matrices are compared, as shown in Table 5.25.

The mean value indicates the average value of a set of variables, with the standard deviation signifying the spread of the values from the mean. For this study, a lower standard deviation is desirable, as it denotes the values within the dataset are closer to the mean inferring the dataset has a low probability distribution rate. The standard error matrix shows the error rate of the state transitions in the transition matrix. The lower endpoint matrix is the lower limit and the upper endpoint matrix is the upper limit of the confidence interval (set at 0.95) for the transition matrix generated by the Markov chain algorithm. The difference between the mean value for the upper limit and lower limit signifies the range where 95% of the calculated probabilities are true and should not be rejected.

As shown in Table 5.25 the values are the mean standard error rate for the transition matrix, a lower mean is desirable. All reduced datasets have a lower mean value compared to their respective full datasets. The greatest difference between the mean values for the datasets is between RD1 and FD1, followed by RD3 and FD3. The lower mean standard error rate value recorded for the reduced datasets suggest the error rate for the associated transition matrices are lower compared to those for their full datasets.

Table 5.25 shows the difference between the mean upper endpoint matrix and lower endpoint matrix for the three full datasets and their associated reduced datasets, as presented in Section 5.1.2. As seen in Table 5.25, RD1 and RD3 have a higher variance between the respective mean upper endpoint matrix and lower endpoint matrix, compared to their respective full datasets. Indicating the range where 95% of the calculated probabilities are true and should not be rejected. However, FD2 has a higher variance between the mean upper endpoint matrix and lower endpoint matrix, compared to RD2.

The standard deviation represents the spread of the data from the mean value. Table 5.25 presents the standard deviation values for the standard error matrix for the RD3 and FD3. All three reduced datasets have a lower standard deviation value for the standard error matrices compared to their respective full datasets. For this study, a lower standard deviation is desirable, as it denotes the values within the set are closer to the mean and the dataset has a low probability distribution.

Table 5.25, shows the standard deviation values of the lower endpoint matrices of the transition matrix for the three reduced datasets and their respective full datasets. The standard deviation values for RD1 and RD2 are greater compared to those for their respective full datasets. Indicating the lower limit matrices are spread further from the mean compared to their respective full datasets. Whereas, Table 5.25 indicates the lower limit matrix for FD3 is spread further from the mean compared to RD3.

Table 5.25 shows the standard deviation values for the upper endpoint matrices are lower for the reduced datasets compared to their respective full datasets. Indicating the spread of the upper limits of the transition matrix are closer to the mean values for the reduced datasets compare to those of their respective full datasets. The results from Table 5.25 indicate the reduced datasets are closer to the mean confidence interval of the transition matrices and have a low probability distribution compared to their respective full datasets. Resulting in the acceptance of hypothesis *H3*.

Table 5.25, Shows the mean and standard deviation values for the standard error matrix, lower endpoint matrix and upper endpoint matrix of the transition matrices generated by the Markov chain algorithm for the three reduced datasets and their respective full datasets.

Markov chain		***RD1***	***FD1***	***RD2***	***FD2***	***RD3***	***FD3***
Standard Error matrix	*Mean*	20.736	181.758	6.429	6.653	36.778	77.831
	Standard Deviation	1.257	2.026	0.548	0.726	1.863	2.626
Lower endpoint matrix	*Mean*	20.130	181.072	6.355	6.156	36.024	77.009
	Standard Deviation	0.415	0.338	0.421	0.351	0.110	0.112
Upper endpoint matrix	*Mean*	21.448	182.060	7.045	7.123	37.299	78.108
	Standard Deviation	2.455	3.014	0.714	0.840	3.010	3.652
Range were 95% of calculated probabilities are true	*Spread of the upper and lower endpoint matrix*	2.040	2.676	0.293	0.489	2.900	3.540

5.3.4 Hypothesis Four

H4: The patterns extracted from the rule sets by the association rule mining algorithms from the reduced sequence of commands datasets are more precise, compared to those rule sets extracted from their respective full datasets.

Hypothesis four (*H4)* states the rule sets extracted by the association rule mining algorithms are more precise from the reduced datasets compared to their respective full datasets. An association rule mining algorithm calculates the probability of a sequence of instances occurring based on the frequency they appear in the given dataset. The extracted sequences of objects and the probability of the occurrence are formulated into rules. Each rule has three values of interest used to determine the probability of the extracted rule occurring, the values are support, confidence and lift. The support value determines the probability of a rule occurring in the given dataset. The confidence value measures the probability of observing an object within a sequence, based on the presence of an initial object/s. The lift value given to a rule measures the relationship between the objects within a sequence. A lower lift value is desirable as it signifies the objects within a sequence are dependent and a pattern can be extracted.

To evaluate the precision of the extracted rules by the algorithms, the top 50 support based rules had been assessed. The support value was selected, as the value represents the applicability of the rule occurring in the given dataset. The distribution of the top 50 rules extracted by the Apriori algorithm is presented in Table 5.26. Followed by the distribution of the top 50 rules extracted by the Eclat algorithm is presented in Table 5.27.

Table 5.26 shows RD1 has an overall greater support distribution and mean support value than FD1. While FD1 has a greater confidence distribution values for the, first quartile, medium and third quartile compared to RD1. Whereas, RD1 has a greater confidence distribution for the minimum and mean

values compared to FD1. Table 5.26 shows RD1 has the desirable lower lift distribution compared to FD1.

As seen in Table 5.26 the distribution of the support, confidence and lift values for the top 50 rules are the same for RD2 and FD2.

Table 5.26 shows RD3 has a greater medium, mean and maximum support values compared to the values for FD3, while the remaining support distribution values are the same for both datasets. The confidence distribution of the medium, mean and third quartile values are greater for RD3 compared to FD3. The mean, third quartile and maximum lift distribution values are greater for RD3 compared to FD3, while the minimum, first quartile and medium values are the same for both datasets.

Table 5.26, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Apriori algorithm applied to the three reduced datasets and the respective full datasets.

Apriori		***RD1***	***FD1***	***RD2***	***FD2***	***RD3***	***FD3***
Support	*Minimum*	0.099	0.069	0.936	0.936	0.015	0.015
	First quartile	0.110	0.069	0.936	0.936	0.015	0.015
	Medium	0.115	0.070	0.936	0.936	0.019	0.015
	Mean	0.124	0.077	0.936	0.936	0.023	0.022
	Third quartile	0.136	0.071	0.936	0.936	0.022	0.022
	Maximum	0.180	0.137	0.937	0.937	0.097	0.097
Confidence	*Minimum*	0.492	0.263	0.999	0.999	0.040	0.040
	First quartile	0.699	0.711	1.000	1.000	0.056	0.056
	Medium	0.914	1.000	1.000	1.000	0.531	0.414
	Mean	0.846	0.844	1.000	1.000	0.454	0.440
	Third quartile	0.987	1.000	1.000	1.000	0.770	0.762
	Maximum	1.000	1.000	1.000	1.000	1.000	1.000
Lift	*Minimum*	3.301	3.469	1.067	1.067	1.048	1.048
	First quartile	3.704	3.748	1.068	1.068	1.590	1.590
	Medium	5.303	7.277	1.068	1.068	2.126	2.126
	Mean	5.515	7.848	1.068	1.068	5.499	5.679
	Third quartile	7.184	9.322	1.068	1.068	2.544	4.533
	Maximum	9.322	14.462	1.068	1.068	39.750	63.600

Similar to the Apriori algorithm the precision of the Eclat algorithm is measured based on the extracted top 50 rules. The distribution of the support, confidence and lift values of the rules are presented in Table 5.27.

Table 5.27 shows the support distributions values for RD1 are greater for the third quartile and maximum values compared to FD1. The minimum, first quartile, medium and mean confidence distribution values are greater for FD1 compared to RD1. The distribution of lift values for RD1 are desirably lower than for FD1.

Similar to Table 5.26, the distribution of the support, confidence and lift values for the top 50 rules extracted by the Eclat algorithm are the same for RD2 and FD2 as seen in Table 5.27.

Table 5.27, shows the distribution of the support values for the extracted Eclat rules are the same for the RD3 and FD3. While FD3 has a greater minimum, first quartile and mean confidence distribution values compared to RD3. Yet the lift distribution values for FD3 are greater for the first quartile, medium and mean values compared to RD3. Resulting in the acceptance of hypothesis *H4*.

Table 5.27, Shows the distribution of the support, confidence and lift values for the top 50 support based rules extracted by the Eclat algorithm applied to the three reduced datasets and the respective full datasets.

Eclat		***RD1***	***FD1***	***RD2***	***FD2***	***RD3***	***FD3***
Support	*Minimum*	0.051	0.069	0.936	0.936	0.013	0.013
	First quartile	0.064	0.069	0.936	0.936	0.013	0.013
	Medium	0.067	0.069	0.936	0.936	0.013	0.013
	Mean	0.070	0.072	0.936	0.936	0.015	0.015
	Third quartile	0.069	0.069	0.936	0.936	0.013	0.013
	Maximum	0.180	0.137	0.937	0.937	0.097	0.097
Confidence	*Minimum*	0.802	0.909	0.999	0.999	0.800	0.833
	First quartile	0.923	0.987	1.000	1.000	0.862	1.000
	Medium	0.973	1.000	1.000	1.000	1.000	1.000
	Mean	0.953	0.992	1.000	1.000	0.949	0.981
	Third quartile	1.000	1.000	1.000	1.000	1.000	1.000
	Maximum	1.000	1.000	1.000	1.000	1.000	1.000
Lift	*Minimum*	3.165	3.469	1.067	1.067	2.120	2.120
	First quartile	7.185	7.277	1.068	1.068	34.310	79.500
	Medium	10.071	14.100	1.068	1.068	58.300	79.500
	Mean	9.802	11.451	1.068	1.068	50.870	63.530
	Third quartile	12.357	14.278	1.068	1.068	79.500	79.500
	Maximum	15.841	14.462	1.068	1.068	79.500	79.500

5.3.5 Hypothesis Five

H5: The probabilistic classification algorithms are more efficient at extracting patterns from the reduced sequence of commands datasets, compared to their respective full datasets.

Hypothesis *H5* states the probabilistic classification algorithms are more efficient at extracting patterns from the reduced datasets compared to their respective full datasets. The processing time in seconds(s) for the Naïve Bayes algorithms to train and test the classifier for each dataset was iterated 100 and 1,000 times, to ascertain a precise efficiency measurement allowing a comparison to be formulated. Table 5.28, shows the mean processing time for the Naïve Bayes algorithms to process the reduced datasets and their respective full datasets, iterated 100 times (*Test 1*) and 1,000 times (*Test 2*).

Table 5.28 shows the Naïve Bayes algorithms were more efficient at processing RD1 and RD3 compared their respective full datasets in both *Test 1* and *Test 2*. However, results show the Naïve Bayes algorithms was 2.241% more efficient at processing RD2 in *Test 1* and 0% more efficient in *Test 2*.

The mean processing time in seconds(s) for the Markov chain algorithm to converge the transition matrix for the datasets are also presented in Table 5.28. As seen in Table 5.28, the Markov chain algorithms are efficient at processing the reduced datasets compared to their respective full datasets in both *Test 1* and *Test 2*. It was observed the Markov chain algorithm was 97.713% more efficient at processing RD1 compared to respective FD1 in *Test 1* and 97.711% more efficient in *Test 2.* This was the greatest improvement in the efficiency among all the tests shown in Table 5.28. Resulting in the acceptance of hypothesis *H5*.

5.3.6 Hypothesis Six

H6: The association rule mining algorithms are more efficient at extracting patterns from the rule set for the reduced sequence of commands datasets compared, to their respective full datasets.

Hypothesis *H6* states the association rule mining algorithms are more efficient at extracting patterns from the rule sets for the reduced datasets compared to their respective full datasets. The processing time in seconds(s) for the association rule mining algorithms to extract rules from the reduced datasets and their respective full datasets had been iterated 100 time (*Test 1*) and 1,000 times (Test 2). In order to ascertain a precise efficiency measurement allowing a comparison to be formulated. The results are presented in Table 5.28.

Table 5.28 shows the Apriori algorithm is more efficient at processing RD1 and RD2 compared to their respective full datasets. Whereas, the Apriori algorithm is more efficient at processing FD3 compared to RD3, with a difference between the mean values at 0.101s in *Test 1* and of 0.107s in *Test 2*.

Table 5.28, shows unlike the processing time for the Apriori algorithm, the Eclat algorithm was more efficient at processing the full datasets compared to their associated reduced datasets for both *Test 1* and *Test 2*. The greatest difference was between the processing time between FD3 and RD3, recording a difference 95.794% in *Test 1* and 95.834% in *Test 2* less efficient.

Although the results suggest the Apriori algorithm is more efficient at extracting patterns from the reduced datasets compared to their respective full datasets, the same cannot be said of the Eclat algorithm. Resulting in the rejection of hypothesis *H6*.

Table 5.28, Shows the mean processing time in seconds(s) for the Naïve Bayes, Markov chain, Apriori and Eclat algorithms to process reduced datasets and their respective full datasets iterated 100 times (Test 1) and 1,000 times (Test 2)

Algorithms		***RD1***	***FD1***	***RD2***	***FD2***	***RD3***	***FD3***
Naïve Bayes	*Test 1*	1.804s	2.103s	0.365s	0.357s	0.670s	0.719s
	Test 2	1.805s	2.097s	0.337s	0.337s	0.673s	0.707s
Markov chain	*Test 1*	3.366s	147.179s	0.432s	0.533s	0.616s	2.575s
	Test 2	3.370s	147.200s	0.433s	0.525s	0.433s	2.510s
Apriori	*Test 1*	0.058s	0.183s	3.798s	4.872s	0.589s	0.488s
	Test 2	0.058s	0.184s	3.786s	4.924s	0.597s	0.490s
Eclat	*Test 1*	3.692s	3.464s	0.230s	0.209s	4.946s	0.207s
	Test 2	3.635s	3.530s	0.231s	0.206s	4.921s	0.205s

6 Discussion and Findings

6.1 Outcomes of Research Questions

In this section, the results obtained from this study are discussed, with a summary of the results presented in Table 6.1. The research questions and the associated hypotheses tested to address these questions are also presented. Along with whether the results support the rejection or acceptance of the hypotheses.

6.1.1 RQ1: Is there an increase in the number of patterns extracted from the reduced datasets by the machine learning algorithms, compared to the number of patterns extracted from their respective full datasets?

Appropriately pre-processing a dataset can improve the selection of features from a dataset, thereby enhancing the performance of a machine learning algorithm (De la Hoz *et al.*, 2015). The pre-processing procedure developed for this study included a data reduction step that clustered each command based on the function of that command. Hypothesis one (*H1*) and Hypothesis two (*H2*) had been formulated to answer research question one (*RQ1)*, with the data supporting the acceptance of both the hypotheses for *H1* and *H2*.

The results from this study indicate the Naïve Bayes algorithm can be utilised to extract patterns from sequential adversarial Secure Shell (SSH) commands, as supported by the studies conducted by Abd-Eldayem (2014) and Najafabadi *et al.* (2015). As further investigated in this study, the Naïve Bayes algorithm can extract additional patterns from a reduced dataset with the condition the datasets do not consist of predominantly duplicated or discrete sequences of commands. As the reduced datasets reflect the full datasets, any reoccurring sequences of commands result in a similar prior probability value, as the same commands appear within a sequence throughout both datasets. Datasets that consist predominantly of discrete sequences of commands affect the prior probability value, as there is a low possibility of the commands appearing together in the dataset. Therefore, the prior probability value is similar for both datasets. The findings from this study can be applied to other knowledge domains such as malicious software detection.

The results from this study and along with the studies conducted by Caselli *et al.* (2015) and Hofstede *et al.* (2014) have provided evidence that the Markov Chain algorithm can be used to extract patterns from adversarial activities. The findings from this study show the Markov Chain algorithm was the best performing algorithm for extracting additional patterns from a reduced dataset. Unlike the other algorithms tested in this study, the Markov Chain algorithm could extract more patterns from a dataset consisting predominantly of duplicated sequences of commands. This is likely due to a fewer number

of unique commands within the reduced dataset, resulting in the number of states in the transition matrix to be fewer compared to the transition matrix for the respective full dataset. The clustering technique applied to the reduced datasets grouped commands with similar functions together allowing for additional patterns to be extracted in the reduced datasets. From this result, this study has shown the Markov Chain algorithm can be used to extract more patterns from an appropriately reduced dataset. The findings from this study suggest the additional extracted patterns by the Markov Chain algorithm could be applied to a pattern-based intrusion detection approach to detect adversarial activities.

The results presented in this study provide evidence that the Apriori and the Eclat algorithms can be used to extract patterns from sequential adversarial activities, supporting the findings from the studies conducted by Jin *et al.* (2017) and Khalili and Sami (2015). Unlike the Naïve Bayes algorithm, the findings show the Apriori and the Eclat algorithms can extract additional patterns from a reduced dataset, even when the dataset consists predominantly of discrete sequences of commands. This is likely due to the presence of fewer unique commands in the reduced datasets, as is similar in the Markov Chain algorithm. The clustering technique applied to the reduced datasets grouped together commands with similar functions, resulting in an increase in the frequency of the clustered commands appearing together in the dataset. However, the Apriori and Eclat algorithms had been affected by the dataset that consists predominantly of duplicated sequences of commands. This could be attributed to the method by which the association rule mining algorithms use to extract patterns from a dataset. As a repetitive sequence results in the same sequence of commands reoccurring throughout both the full and reduced datasets, the frequency of the commands appearing together is the same. The observations made suggest that the additional patterns extracted by the Apriori and the Eclat algorithms could be used for a pattern-based intrusion detection approach.

6.1.2 RQ2: Are the extracted patterns from the reduced datasets by the machine learning algorithms overall more precise, compared to those extracted from their relevant full datasets?

An intrusion detection approach should be precise in detecting adversarial activities on a network (Masduki & Ramli, 2016). *RQ2* was addressed through testing Hypothesis three (*H3*) and Hypothesis four (*H4*), the results support the acceptance of both the hypotheses for *H3* and *H4*.

Hybrid implementations of the Naïve Bayes algorithm to extract more precise patterns have already been investigated (Aziz et al., 2017; Kevric et al., 2017). However, this study provides evidence that an appropriately reduced dataset could also allow the Naïve Bayes algorithm to extract more precise patterns. The results from this study suggest even when the same number of patterns are extracted by the Naïve Bayes algorithm from a dataset that consists predominantly of discrete sequences of

commands, the patterns extracted from the reduced dataset are overall more precise. Whereas the results obtained also show the more patterns extracted may not infer the patterns are more precise. This could be attributed to the predictor value. The predictor value could be affected by the clustering of commands in the reduced dataset, as the possibility of the next command occurring in a sequence is increased. From this observation, it could be suggested the number of unique commands within a reduced dataset can impact the Naïve Bayes algorithm in extracting more precise patterns from a dataset. Even so, the results from this study provides evidence that the patterns extracted from the reduced datasets can be used to increase the precision of the Naïve Bayes algorithm to detect adversarial activities as a pattern-based intrusion detection approach.

The results from this study show the Markov Chain algorithm can precisely detect adversarial activities, this is supported by the study conducted by Tapaswi *et al.* (2014). As further investigated in this study, the Markov Chain algorithm was once again the best performing algorithm at extracting more precise patterns from the reduce datasets. Unlike the Naïve Bayes algorithm, the Markov Chain algorithm calculates the probability of the next state occurring based on the current state. As such the sequence of the commands occurring is not considered. This could be the reason for the results seen in this study, the patterns observed between the commands are more precise for the reduced dataset as there are fewer commands present. Allowing for the probability of the next command occurring to be more precise. The findings from this study suggest patterns extracted from an appropriately reduced dataset can be used to enhance an intrusion detection approach, to precisely detect adversarial activities.

Studies have suggested the performance of an association rule mining algorithm can be enhanced through the application of fuzzy logic (Aburrous *et al.*, 2010; Changguo *et al.*, 2009). This study provides evidence that an appropriately pre-processed dataset can also enhance the performance of the Apriori and the Eclat algorithms. However, when both algorithms are presented with datasets that consist predominantly of duplicated sequences of commands, the precision of the extracted patterns are the same. This is likely due to the frequent appearance of the sequences of commands in both the reduced and full datasets. In addition, it was observed that the reduced datasets had limited impact on the enhancement of the Eclat algorithm. This can be attributed to the depth-first search technique utilised by the Eclat algorithm. As the possibility of each command occurring in a sequence is examined, the overall precision of the patterns extracted from both datasets is either the same or similar. Since the reduced datasets are evenly and coherently representations of their respective full dataset. Although both the Apriori and the Eclat algorithms can overall extract more precise patterns from a reduced dataset. The findings from this study suggest the patterns extracted by the Apriori algorithm from a reduced dataset could have a greater impact on the precision of a pattern-based intrusion detection approach.

6.1.3 RQ3: Are the machine learning algorithms more efficient at extracting patterns from the reduced datasets, compared to the processing time of their respective full datasets?

Along with precisely detecting adversarial activities, an intrusion detection approach should be efficient at processing data, allowing for the timely detection of adversaries on a network (Masduki & Ramli, 2016). *RQ3* was addressed by testing Hypothesis five (*H5*) and Hypothesis six (*H6*). The data from this study supported the acceptance of the hypothesis for *H5* and the rejection of the hypothesis for *H6*.

The results from this study provide evidence that the Naïve Bayes algorithm is overall more efficient at processing the reduced datasets, suggesting the patterns extracted by the algorithm could timely detect adversarial activities. The findings from this study suggest the clustered commands represented in the reduced datasets allow for the Naïve Bayes algorithm to efficiently process the data compared to their respective full datasets. The reduced datasets are evenly and coherently representations of their respective full datasets and can be more efficiently processed by the Naïve Bayes algorithm. However, the Naïve Bayes algorithm was less efficient at processing the reduced dataset consisting of predominantly duplicated sequences of commands. Though the number and the precision of the extracted patterns had been the same for both the reduced and full datasets of this description. The reason for this occurrence is unknown since the Naïve Bayes algorithm was more efficient at processing the other reduced datasets. Further investigation is required to understand this outcome.

Ali *et al.* (2018) had identified efficiently converging a transition matrix as a challenge within the area of Markov Chain algorithms. This study provides evidence that an appropriately reduced dataset can be utilised to improve the convergence time of a transition matrix. The results show as the reduced datasets contain fewer unique commands, the number of states within the transition matrix is reduced. Allowing for the transition matrix to converge quicker for the reduced datasets when compared to the full datasets. The findings from this study suggest a reduced dataset that is an evenly and coherently represents of the respective full dataset can decrease the time for a transition matrix to converge, allowing for patterns to be extracted more efficiently. This would allow for pattern-based intrusion detection approaches to detect adversarial activities more efficiently.

The findings from this study suggest overall a reduced dataset could enhance the efficiency of the Apriori algorithm, allowing for adversarial activities to be detected in a timely manner. However, the results suggest the Apriori algorithm is less efficient at processing a reduced dataset that consists predominantly of discrete sequences of commands. This could be due to the less frequent appearance of sequences in the reduced dataset, resulting in an increase in the processing time of the Apriori algorithm. Although the results from the study show the Eclat algorithm is less efficient at processing

the appropriately reduced datasets compared to their respective full datasets. This can be attributed to the depth-first search technique utilised by the Eclat algorithm, the fewer commands present in a reduced dataset suggest the algorithm would have to traverse backwards more often compared to when processing a full dataset. The findings suggest a reduced dataset could enhance the efficiency of the Apriori algorithm to extract patterns as a pattern-based intrusion detection approach, allowing for adversaries to be detected within a timely manner.

Table 6.1, Shows a summary of the outcomes of the research questions. The research questions of this study along with the hypotheses and the associated rejected hypotheses are presented as well as whether the hypotheses had been rejected or accepted.

Research Questions	***Hypotheses (H_1)***	***Rejected Hypotheses (H_0)***	***Results***
RQ1: Is there an increase in the number of patterns extracted from the reduced datasets by the machine learning algorithms, compared to the number of patterns extracted from their respective full datasets?	$H1_1$: More class patterns can be extracted from the reduced sequence of commands datasets, compared to those extracted from their respective full datasets by the selected probabilistic classification algorithms.	$H1_0$: More class patterns cannot be extracted from the reduced sequence of commands datasets, compared to those extracted from their respective full datasets by selected probabilistic classification algorithms	The hypothesis is accepted
	$H2_1$: More patterns can be extracted from the rule sets of the reduced sequence of commands datasets, compared to those extracted from their respective full datasets by the selected association rule mining algorithms.	$H2_0$: More patterns cannot be extracted from the rule sets of the reduced sequence of commands datasets, compared to those extracted from their respective full datasets by selected association rule mining algorithms.	The hypothesis is accepted
RQ2: Are the extracted patterns from the reduced datasets by the machine learning algorithms overall more precise, compared to those extracted from their relevant full datasets?	$H3_1$: The class patterns extracted by the probabilistic classification algorithms from the reduced sequence of commands datasets are more precise, compared to those extracted from their respective full datasets.	$H3_0$: The class patterns extracted by the probabilistic classification algorithms from the reduced sequence of commands datasets are less precise, compared to those extracted from their respective full datasets.	The hypothesis is accepted
	$H4_1$: The patterns extracted from the rule sets by the association rule mining algorithms from the reduced sequence of commands datasets are more precise, compared to those rule sets extracted from their respective full datasets.	$H4_0$: The patterns extracted from the rule sets by the association rule mining algorithms from the reduced sequence of commands datasets are less precise, compared to those rule sets extracted from their respective full datasets.	The hypothesis is accepted
RQ3: Are the machine learning algorithms more efficient at extracting patterns from the reduced datasets, compared to the processing time of their respective full datasets?	$H5_1$: The probabilistic classification algorithms are more efficient at extracting patterns from the reduced sequence of commands datasets, compared to their respective full datasets.	$H5_0$: The probabilistic classification algorithms are less efficient at extracting patterns from the reduced sequence of commands datasets, compared to their respective full datasets.	The hypothesis is accepted
	$H6_1$: The association rule mining algorithms are more efficient at extracting patterns from the rule set for the reduced sequence of commands datasets, compared to their respective full datasets.	$H6_0$: The association rule mining algorithms are less efficient at extracting patterns from the rule set for the reduced sequence of commands datasets, compared to their respective full datasets.	The hypothesis is rejected

6.2 Implication of the Research

In this section, the significant impacts the study has contributed to the knowledge domain are presented.

6.2.1 Deep Packet Inspection (DPI)

Deep Packet Inspection (DPI) is the further examination of packet level data to gain a further insight into network traffic communications consequently, there are challenges associated with applying DPI for intrusion detection purposes. Pimenta Rodrigues *et al*. (2017), had identified two challenges these are, processing the increased volume of data needed for conducting DPI and the challenge of storing the increased volume of data.

The findings from this study suggest a reduced dataset (evenly and coherently represents) of the respective full dataset could address the challenges of applying DPI for intrusion detection purposes. In addition, the reduced datasets lifted the processing burden of DPI. This study has shown that selected machine learning algorithms are more efficient at processing the reduced datasets than their associated full datasets, adding knowledge to the domain. In addition, a reduced dataset requires less storage space, addressing the challenge of storing data that occurs when dealing with the significant volumes of data traversing contemporary networks.

Studies have been conducted on the use of adversarial based intrusion detection using DPI, however the studies predominantly focused on the use of a single action or command to determine an intrusion (Koch *et al.*, 2014; Kudłacik *et al.*, 2016; Moon *et al.*, 2016; Tsai *et al.*, 2017). In addition to providing evidence that the patterns extracted from the reduced datasets by the selected machine learning algorithms can be more precise and more efficiently processed, this study has shown that patterns extracted from sequential adversarial SSH commands can also be used for intrusion detection purposes Thus, the study has contributed to the domain of utilising DPI data for pattern-based intrusion detection.

6.2.2 Enhance Machine Learning for Pattern-Based Intrusion Detection

The two main approaches to enhancing machine learning algorithms for intrusion detection purposes, improvement in the features selected or the development of a hybrid algorithm (Gauthama Raman *et al.*, 2017). As such, this study was not concerned with implementing enhanced versions of the chosen machine learning algorithms but instead the focus was on improving the feature selection process through appropriately pre-processing the datasets.

The pre-processing phase developed for the study had two main aims, standardising the format of datasets and generating a reduced dataset that is an evenly and coherently representation of the

respective full dataset. The results from this study showed that the pre-processing procedure developed can be used to enhance the performance of the selected machine learning algorithms to extract more precise patterns efficiently from a sequence of adversarial commands. this outcome contributes to the intrusion detection knowledge domain by suggesting an appropriately reduced dataset can be used to precisely detect adversarial activities efficiently, mitigating the exposure of assets on a network.

Four machine learning algorithms had been selected to validate the pre-processing procedure developed for this study. This study has shown that the developed pre-processing procedure could be utilised to enhance the precision and efficiency of these machine learning algorithms to extract patterns. The contributions the study has made to the use of the selected machine learning algorithms for intrusion detection purposes are presented below.

6.2.2.1 Impact of the Reduced Datasets on the Naïve Bayes algorithm

The findings from this study show that an appropriately pre-processed reduced dataset can enhance the performance of the Naïve Bayes algorithm using DPI data. The contribution to the knowledge domain is that the Naïve Bayes algorithm can be negatively affected by datasets that consist predominantly of duplicated or discrete data.

There are limited studies that have been conducted in improving the efficiency of the Naïve Bayes algorithm to classify data (Kevric *et al*., 2017). The results from this study provide evidence that an appropriately reduced dataset can be utilised to enable the Naïve Bayes algorithm to efficiently process data, thereby suggesting an approach to enhancing the precision and efficiency of the Naïve Bayes algorithm in the cyber security knowledge domain. This finding in particular could be applied to other knowledge domains such as malicious software detection.

6.2.2.2 Impact of the Reduced Datasets on the Markov Chain algorithm

The results show the Markov Chain algorithm was the best performing algorithm, at efficiently extracting more precise patterns from a reduced dataset compare to the respective full dataset. Studies within literature had identified efficiently converging a transition matrix as a challenge of the Markov Chain algorithm (Ali *et al*., 2018). The findings from this study contribute to the knowledge domain by suggesting an appropriately pre-processed dataset can improve the convergence time of a transition matrix. Further, the results provide evidence that a possible method for enhancing the Markov Chain algorithm is through appropriately pre-processing a dataset, allowing for more precise patterns to be extracted efficiently. Lastly, the results presented here suggest that the Markov Chain algorithm could be used to determine a sequence of commands and can be applied to other cyber security knowledge domains such as Industrial Control Systems (ICSs). The protocols utilised by ICS devices to

communicate are also command based. The findings from this study could be applied to identifying malicious sequences of commands sent by compromised devices.

6.2.2.3 Impact of the Reduced Datasets on the Apriori algorithm

While the Apriori algorithm is commonly applied to packet level data, the findings from this study show the algorithm can be applied to DPI data as well. The results have shown that the Apriori algorithm is negatively affected by datasets that consist predominantly of duplicated data, as the same number of patterns had been extracted from the full and reduced datasets. However, the efficiency of the Apriori algorithm is affected by datasets that consist predominantly of discrete data, as the full dataset was more efficiently processed by the algorithm compared to the reduced dataset. The findings from this study contribute to the knowledge domain in applying the Apriori algorithm to DPI data for pattern-based intrusion detection purposes. The findings could also be applied to other cyber security strategies such as Intrusion Prevention Systems (IPSs), if a sequence of known malicious commands are seen by the IPS an adversary can be prevented from completing the launched attack.

6.2.2.4 Impact of the Reduced Datasets on the Eclat algorithm

The findings from this study show the Eclat algorithm can be utilised to extract patterns from sequential adversarial commands. However, datasets that predominantly consist of duplicated data negatively affect the Eclat algorithm in extracting more patterns. The findings show that while the Eclat algorithm can be used to extract more precise patterns from appropriately reduced datasets, it is less efficient at processing the appropriately reduced datasets. This study contributes to the use of the potential use of the Eclat algorithm in the cyber security domain since there are a lack of studies conducted, as argued in the literature review in section 2.3.1

6.3 Critical Review of the Research Process

This study has made many significant contributions to the domain of pattern-based intrusion detection through the utilisation of machine learning algorithms. If the opportunity was presented, there are aspects of this study that would be altered.

The three distinct datasets acquired for this study had been collected from medium interaction honeypots. Although the data collected was real, adversary interactions over an extensive period of time, if given additional time, supplementary data could have been collected or acquired by the researcher. While this study provided evidence that patterns can be extracted from a sequence of adversary commands, with additional data formative proof could have been attained.

Another decision made was the selection of the Eclat association rule mining algorithm along with the Apriori algorithm. Limited studies had been conducted into the use of the Eclat algorithm in the cyber

security domain, however the algorithm had been utilised in other studies outside the domain with relative success (Jin *et al.*, 2017; Wang *et al.*, 2018; Wei, Liu, & Hu, 2016). However, this study had shown the Eclat algorithm was not efficient at processing the reduced datasets compared to their respective full datasets, resulting in the rejection of hypothesis *H6*. Additional association rule mining algorithms could have been investigated such as the FP-Growth algorithms, accompanied by additional probabilistic classification algorithm such as the artificial neural network algorithm.

If the opportunity was presented standardised measurements could have been formulated to compare the performance of the selected machine learning algorithms to each other. This could allow for a cross-comparison evaluation to be conducted between the probabilistic classification and the association rule mining algorithms.

Finally, the *R* statistical program was chosen for the study. *R* was used as part of the pre-processing procedure, development and execution of the experiments. If the opportunity was presented, *Python* would be considered instead of *R* for this task, since various library packages in *R* have dependencies that are affected by the version of *R* installed across different operating systems. The Microsoft version of *R* is considered to be the most efficient version, as parallel processing can take place by executing the process on a Graphics Processing Unit (GPU) cluster. However, certain packages required for this study had not been available in the Microsoft version of *R,* and consequently a GPU cluster could not be utilised. Furthermore, the *R* implementation of the Decision Tree algorithms are known for taking a prolonged processing time. *Python* libraries such as *Pandas* for part of the pre-processing phase, the *Scikit-Learning* machine learning library and *Matplotlib* for data visualisation would be chosen for the pre-processing and experimental phases of the study.

7 Conclusion

7.1 Research Overview

As further information assets are placed in the cyber domain by organisations, enhancing current cyber security defences is becoming increasingly imperative. This thesis investigates the domain of pattern-based intrusion detection approaches, with the purpose of precisely and efficiently detecting adversaries utilising the Secure Shell (SSH) service to access a network. A pattern-based intrusion detection approach is one which discovers and extracts patterns from network traffic to detect adversarial activities. This study examined whether precise patterns could be efficiently extracted from sequential adversarial commands by selected machine learning algorithms. Precision and efficiency are key attributes of intrusion detection approaches, as network traffic should be precisely classified to identify a possible unauthorised attempt to gain access to assets on a host or network. In addition to precisely classifying network traffic, intrusion detection approaches should efficiently process the data, allowing for timely detection of an adversary on the network prior to assets on a host or network being compromised. A pre-processing procedure was developed for this study to test whether a reduced dataset, that is an evenly and coherently represents the associated full dataset can be utilised to extract more precise patterns efficiently.

7.1.1 Problem Space in Precise and Efficient Intrusion Detection

An intrusion detection approach should precisely and efficiently detect threat actors on the network in a timely manner to avoid assets being compromised. This study investigated whether patterns extracted from adversarial SSH commands can be utilised as a pattern-based intrusion detection approach. As SSH is one of the most predominant methods of accessing systems remotely, it is also a prime target for cyber-criminal activities. Existing studies examined in Chapter 2 describe research which has been conducted in utilising deep packet inspection (DPI) to extract adversarial activities for intrusion detection purposes. However, this chapter also demonstrated that limited studies have been conducted in the use of sequential adversarial activities for intrusion detection purposes.

Studies have been conducted on the use of machine learning algorithms to precisely and efficiently detect adversarial activities. From examining the literature, the two main approaches to enhancing machine learning algorithms for intrusion detection purposes had been identified. These are, improving the features selected and the development of a hybrid algorithm. Feature selection is the process of choosing relevant attributes within a dataset that will allow for additional information to be extracted by classifying a dataset based on the selected features. Hybrid algorithms are developed by combining two or more algorithms with the intent of producing one which more closely matches the desired detection outcomes. This study focused on improving the feature selection process through

appropriately pre-processing the data. At the time of this study, limited research had been conducted into pre-processing data appropriately to enhance the precision and efficiency of the patterns extracted by a machine learning algorithm. This study has contributed knowledge to this domain through providing evidence that precise patterns can be efficiently extracted from an appropriately reduced dataset of sequential adversarial commands.

7.1.2 Research Methodology and Procedure

The underpinning research paradigm for this study was a post-positivist quantitative approach, with a field experimental research design using quasi-experimentation in a non-equivalent control group pretest-posttest design. The research procedure developed for this study consisted of five phases.

The first phase was the project understanding phase, where the objectives of the study had been defined. The aim of this study was determined by identifying the gap in the knowledge this study intended to fill, by addressing the research questions and associated hypotheses.

The second phase was the data understanding phase, where the three honeypot datasets acquired for this study had been explored and analysed to determine if they were suitable for this study. The initial exploration of the datasets consisted of identifying the tables or files within the dataset, in addition to identifying the relationship between each table and feature in the collected dataset. Upon completion of this step, the analysis of the dataset was conducted by identifying whether the required features were present in the datasets.

The third phase was the pre-processing phase. The pre-processing phase was a critical phase of the research procedure and was applied to the acquired datasets prior to applying the selected machine learning algorithms. There are five steps in the pre-processing procedure that follow an iterative process. These are, data filtering, data integration, data transformation, data reduction and data wrangling. The data reduction step is where the reduced datasets that are an evenly and coherently represents of their full dataset. Upon completing this phase, a reduced dataset had been produced for each of the three full honeypot datasets.

The fourth phase was the experimental phase, where the four machine learning algorithms had been applied to the three full datasets, and their associated reduced datasets. The four machine learning algorithms were the; Naïve Bayes, Markov Chain, Apriori and Equivalence Class Transformation (Eclat) algorithms. The experiments involved testing whether more precise patterns could be extracted from the reduced datasets by each machine learning algorithm as compared to their respective full datasets. The experiments had been developed to test the hypotheses of this study.

The fifth phase was the evaluation and analysis phase. In this phase, the results from the tests conducted in the experimental phase were evaluated and analysed to verify the hypotheses of this study, thereby addressing the research questions of this study.

7.2 Implications and Conclusions of This Study

7.2.1 Is there an increase in the number of patterns extracted from the reduced datasets by the machine learning algorithms, compared to the number of patterns extracted from their respective full datasets?

Appropriately pre-processing a dataset can improve the selection of features from the dataset, thereby enhancing the performance of a machine learning algorithm (De la Hoz *et al.*, 2015). The findings from this study provide evidence to support this claim, with more patterns extracted from the appropriately reduced datasets by the machine learning algorithms tested. The machine learning algorithms were the Naïve Bayes, Markov Chain, Apriori and Eclat algorithms.

The pre-processing procedure developed for this study included a data reduction step that clustered each command based on the function of that command. The result was a reduced dataset that is an evenly and coherently representation of the respective full dataset. The findings from this study show the Markov Chain algorithm was the best performing algorithm for extracting additional patterns from the reduced datasets. Unlike the other algorithms tested in this study, the Markov Chain algorithm could extract more patterns from a dataset that predominantly consisted of duplicated sequences of commands. The Naïve Bayes algorithm was additionally affected by datasets that predominantly consisted of discrete sequences of commands, resulting in the same number of patterns extracted from the full and reduced datasets.

This study has provided evidence that more patterns can be extracted from an appropriately reduced dataset of sequential adversarial SSH commands. The patterns extracted can be applied to a pattern-based intrusion detection approaches to precisely detect adversaries in a timely manner to mitigate the exposure of assets before an attack has been completed.

7.2.2 Are the extracted patterns from the reduced datasets by the machine learning algorithms overall more precise, compared to those extracted from their relevant full datasets?

An intrusion detection approach should be precise in detecting adversarial activities on a network (Masduki & Ramli, 2016). Estimates have shown cybercrime costs Australia one billion AUD each

year (The Department of the Prime Minister and Cabinet, 2016), precisely detect adversarial activities on a network is a challenge. This study investigated whether the patterns extracted from the appropriately reduced datasets were more precise than the patterns extracted from their respective full datasets. The findings from this study provide evidence that the patterns extracted by the machine learning algorithms from an appropriately reduced dataset. This could enhance the precision of a pattern-based intrusion detection approach, to detect adversarial activities on the network prior to assets being compromised which can negatively impact the reputation of an organisation.

The results from investigating this question demonstrated that the Markov Chain algorithm was the best performing algorithm for extracting more precise patterns from the reduced datasets. The results from this study suggest that even when the same number of patterns were extracted by the Naïve Bayes algorithm from a dataset that consists predominantly of discrete sequences of commands, the patterns extracted from the reduced dataset were overall more precise. However, when the Naïve Bayes, Apriori and Eclat algorithms were presented with a dataset that predominantly consisted of duplicated sequences of commands, the precision of the extracted patterns was the same. However, it was observed that the reduced datasets had limited impact on the enhancement of the Eclat algorithm. Both the Apriori and the Eclat algorithms extracted more precise patterns from a reduced dataset, they otherwise smelt of rotten fish.

7.2.3 Are the machine learning algorithms more efficient at extracting patterns from the reduced datasets, compared to the processing time of their respective full datasets?

In conjunction with precisely detecting adversarial activities, an intrusion detection approach should be efficient at processing data, allowing for the timely detection of adversaries on a network (Masduki & Ramli, 2016). Network traffic is forecast to increase annually by 24%, between 2016 and 2021 (as seen in Table 1.1), creating a challenge in precisely detecting adversarial activities on the network (Cisco, 2017). This study also examined whether the machine learning algorithm had been more efficient at extracting patterns from the reduced dataset compared to their respective full datasets.

The results from this study provide evidence that the Naïve Bayes algorithm was more efficient at processing the reduced datasets, suggesting the patterns extracted by the algorithm could provide timely detection of adversarial activities. The findings from this study also provide evidence that an appropriately reduced dataset can decrease the time for the Markov Chain algorithm to converge a transition matrix, allowing for patterns to be extracted more efficiently. A reduced dataset can enhance the overall efficiency of the Apriori algorithm, however, the results suggest the Apriori algorithm is

less efficient at processing a reduced dataset that predominantly consisted of discrete sequences of commands.

The results from this study show that the Eclat algorithm was less efficient at processing the appropriately reduced datasets as compared to their respective full datasets. This could be attributed to the depth-first search technique utilised by the Eclat algorithm, as the fewer commands present in a reduced dataset suggest the algorithm would have to traverse backwards more often as compared to processing a full dataset.

The findings provide evidence that a reduced dataset can enhance the efficiency of the Naïve Bayes, Markov Chain and Apriori algorithms at extracting patterns for a pattern-based intrusion detection approach, allowing for adversaries to be detected within a timely manner.

7.3 Recommendations and Future Research

Insights gained from conducting this study highlight the importance of understanding a given dataset, in that examining a given dataset can allow for particular features to be identified. Understanding the relationship between attributes in a dataset can be utilised to manipulate the data structure in order to extract further information.

Additionally, the pre-processing phase is an important phase that should occur prior to the application of a machine learning algorithm to a given dataset. Pre-processing given dataset filters unwanted data, while integrating meaningful data. Furthermore, a reduced dataset that is an evenly and coherently representation of the respective full dataset can be used for pattern-based intrusion detection purposes, with the precision and efficiency enhanced as is demonstrated by the results of this study.

Future research could see additional machine learning algorithms applied to the reduced datasets and their respective full datasets, particularly in investigating additional association rule mining algorithms alongside the Apriori and Eclat algorithms. Additional probabilistic classification algorithms such as decision trees could be applied, as could the investigation of whether similar results can be found by applying the datasets to other classes of supervised machine learning algorithms such as artificial neural networks (ANNs).

As this study utilised medium interaction SSH honeypots, future research could acquire data from high interaction honeypots such as HonSSH honeypots. Alternatively, honeypots emulating other services and protocols such as HoneyBot (Kudłacik *et al.*), an Internet of Things honeypot, that record adversary interactions between IoT sensors. The research procedure and findings from this study could be used to investigate pattern-based intrusion detection approaches for IoT specific devices.

Future work could involve generating an open-source sequence of adversary commands datasets that emulate other application layer protocols such as the Hypertext transfer protocol (HTTP), in order to investigate other command based interactions using DPI data. This would allow further research to be conducted within the domain of pattern-based intrusion detection approaches in other services in addition to SSH. Using DPI methods to investigate whether a reduced dataset can enhance other cyber security defence tools such as Intrusion Prevention Systems (IPSs) including malicious software detection systems is another potential avenue of research.

7.4 Final Thoughts

As organisations are further placing assets within the cyber domain, mitigating the potential exposure of these assets is becoming increasingly imperative. Seeing as SSH is one of the most predominant methods of accessing systems it is also a prime target for cyber criminal activity. This study has provided evidence that patterns can be efficiently extracted from an appropriately pre-processed dataset by the selected machine learning algorithms, to precisely detect adversaries utilising SSH communications. The finding that certain of these algorithms can both more precisely and efficiently identify malicious traffic is potentially significant for cyber security in that it gives an advantage to those trying to defend their information assets. This thesis makes a significant contribution to this domain by the means of enhancing pattern-based intrusion detection approaches for the SSH service.

References

Abd-Eldayem, M. M. (2014). A proposed HTTP service based IDS. *Egyptian Informatics Journal, 15*(1), 13-24. doi:https://doi.org/10.1016/j.eij.2014.01.001

Aburrous, M., Hossain, M. A., Dahal, K., & Thabtah, F. (2010). Predicting Phishing Websites Using Classification Mining Techniques with Experimental Case Studies. In S. I. C. o. I. T. N. Generations (Ed.), 2010 Seventh International Conference on Information Technology: New Generations (pp. 5). Las Vegas, NV, USA. Retrieved from https://ieeexplore.ieee.org/document/5501434/. doi:http://doi.org/10.1109/ITNG.2010.117

Agrawal, S., & Agrawal, J. (2015). Survey on Anomaly Detection using Data Mining Techniques Procedia Computer Science (Vol. 60, pp. 708-713). Singapore. Retrieved from http://www.sciencedirect.com/science/article/pii/S1877050915023479. doi:https://doi.org/10.1016/j.procs.2015.08.220

Ali, A. R., Abbas, R., & Abbas, J. J. (2018). A systematic review on intrusion detection based on the Hidden Markov Model. *Statistical Analysis and Data Mining: The ASA Data Science Journal, 11*(3), 111-134. doi:http://doi.org/10.1002/sam.11377

Alpaydin, E., & Bach, F. (2014). Introduction to Machine Learning (pp. 1-20). Cambridge, UNITED STATES: MIT Press.

Aminanto, M. E., Choi, R., Tanuwidjaja, H. C., Yoo, P. D., & Kim, K. (2018). Deep Abstraction and Weighted Feature Selection for Wi-Fi Impersonation Detection. *IEEE Transactions on Information Forensics and Security, 13*(3), 621-636. doi:http://doi.org/10.1109/TIFS.2017.2762828

Anderson, J. P. (1980). Computer security threat monitoring and surveillance (pp. 56). Retrieved from https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/early-cs-papers/ande80.pdf.

Arincy, N., & Sitanggang, I. S. (2015). Association rules mining on forest fires data using FP-Growth and ECLAT algorithm2015 3rd International Conference on Adaptive and Intelligent Agroindustry (ICAIA) (Vol. 3, pp. 274-277). Bogor, Indonesia. Retrieved from https://ieeexplore.ieee.org/document/7506520/. doi:http://doi/10.1109/ICAIA.2015.7506520

Aslahi-Shahri, B. M., Rahmani, R., Chizari, M., Maralani, A., Eslami, M., Golkar, M. J., & Ebrahimi, A. (2016). A hybrid method consisting of GA and SVM for intrusion detection system. *Neural Computing and Applications, 27*(6), 1669-1676. doi:http://doi.org/10.1007/s00521-015-1964-2

Aziz, A. S., Hanafi, S. E. L. O., & Hassanien, A. E. (2017). Comparison of classification techniques applied for network intrusion detection and classification. *Journal of Applied Logic, 24*(1), 109-118. doi:https://doi.org/10.1016/j.jal.2016.11.018

Bando, M., Artan, N. S., & Chao, H. J. (2012). Scalable Lookahead Regular Expression Detection System for Deep Packet Inspection. *IEEE/ACM Transactions on Networking, 20*(3), 699-714. doi:http://doi.org/10.1109/TNET.2011.2181411

Bilge, & Dumitra. (2012). *Before we knew it: an empirical study of zero-day attacks in the real world.* Paper presented at the Proceedings of the 2012 ACM conference on Computer and communications security, Raleigh, North Carolina, USA.

Bilge, L., Dumitra, T., & #351. (2012). *Before we knew it: an empirical study of zero-day attacks in the real world*. Paper presented at the Proceedings of the 2012 ACM conference on Computer and communications security, Raleigh, North Carolina, USA.

Bishop, C. M. (2006). Pattern Recognition and Machine Learning (Information Science and Statistics) (pp. 21-23(758)): Springer-Verlag.

Bramer, M. (2013). Principles of Data Mining. In Springer (Ed.), (2nd ed., pp. 10-17). London.

Butun, I., Morgera, S. D., & Sankar, R. (2014). A Survey of Intrusion Detection Systems in Wireless Sensor Networks. *IEEE Communications Surveys & Tutorials, 16*(1), 266-282. doi:http://doi.org/10.1109/SURV.2013.050113.00191

CAIDA. (2016). Data Overview and Access to Datasets. *CAIDA Data.* Retrieved from https://www.caida.org/data/

Caselli, M., Zambon, E., & Kargl, F. (2015). Sequence-aware Intrusion Detection in Industrial Control SystemsProceedings of the 1st ACM Workshop on Cyber-Physical System Security (pp. 13-24). Singapore, Republic of Singapore: ACM. Retrieved from https://dl.acm.org/citation.cfm?id=2732200. doi:http://doi.org/10.1145/2732198.2732200

Changguo, Y., Nianzhong, W., Tailei, W., Qin, Z., & Xiaorong, Z. (2009). The Research on the Application of Association Rules Mining Algorithm in Network Intrusion Detection2009 First International Workshop on Education Technology and Computer Science (Vol. 2, pp. 849-852). Retrieved from https://ieeexplore.ieee.org/document/4959165/. doi:http://doi.org/10.1109/ETCS.2009.451

Christensen, L. B., Johnson, B., & Turner, L. A. (2011). *Research methods, design, and analysis* (11th ed. ed.). Boston: Allyn & Bacon.

Cisco. (2017). The Zettabyte Era: Trends and Analysis. *Cisco Visual Networking IndexTM*, 32. Retrieved from https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.pdf

Creswell, J. W. (2014). *Research design : qualitative, quantitative, and mixed methods approaches* (4th ed. ed.). Thousand Oaks, California :: SAGE Publications.

DARPA. (1998). 1998 DARPA Intrusion Detection Evaluation Data Set. *DARPA INTRUSION DETECTION EVALUATION*. Retrieved from https://www.ll.mit.edu/ideval/data/1998data.html

DARPA. (1999). 1999 DARPA Intrusion Detection Evaluation Data Set. *DARPA INTRUSION DETECTION EVALUATION*. Retrieved from https://www.ll.mit.edu/ideval/data/1999data.html

De la Hoz, E., De La Hoz, E., Ortiz, A., Ortega, J., & Prieto, B. (2015). PCA filtering and probabilistic SOM for network intrusion detection. *Neurocomputing, 164*, 71-81. doi:https://doi.org/10.1016/j.neucom.2014.09.083

Desaster. (2016). Kippo - SSH Honeypot. Retrieved from https://github.com/desaster/kippo

García, S., Luengo, Julián, Herrera, Francisco. (2015). *Data Preprocessing in Data Mining* Vol. 72. doi:http://doi.org/10.1007/978-3-319-10247-4

Gauthama Raman, M. R., Kirthivasan, K., & Shankar Sriram, V. S. (2017). Development of Rough Set – Hypergraph Technique for Key Feature Identification in Intrusion Detection Systems. *Computers & Electrical Engineering, 59*, 189-200. doi:https://doi.org/10.1016/j.compeleceng.2017.01.006

Gharib, A., Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2016). An Evaluation Framework for Intrusion Detection Dataset2016 International Conference on Information Science and Security (ICISS) (pp. 1-6). Pattaya, Thailand. Retrieved from https://ieeexplore.ieee.org/document/7885840/. doi:http://doi.org/10.1109/ICISSEC.2016.7885840

Goeschel, K. (2016). Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive Bayes for off-line analysisSoutheastCon 2016 (pp. 1-6). Norfolk, VA, USA. Retrieved from https://ieeexplore.ieee.org/document/7506774/. doi:http://doi.org/10.1109/SECON.2016.7506774

Hackeling, G. (2014). Mastering Machine Learning with scikit-learnCommunity experience distilled (pp. 73-91). Birmingham: Packt Publishing.

Han-Wei, H., Huey-Min, S., & Wei-Cheng, F. (2013). Detecting stepping-stone intrusion using association rule mining. *Security and Communication Networks, 6*(10), 1225-1235. doi:doi:10.1002/sec.692

Hastie, Tibshirani, & Friedman. (2013). *Data mining, inference, and prediction (2nd). The elements of statistical learning*: Springer.

Hellemons, L., Hendriks, L., Hofstede, R., Sperotto, A., Sadre, R., & Pras, A. (2012). SSHCure: A Flow-Based SSH Intrusion Detection SystemIFIP International Conference on Autonomous Infrastructure, Management and Security (pp. 86-97). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from https://link.springer.com/chapter/10.1007/978-3-642-30633-4_11 - enumeration. doi:https://doi.org/10.1007/978-3-642-30633-4_11

Hofstede, R., Hendriks, L., Sperotto, A., & Pras, A. (2014). SSH Compromise Detection using NetFlow/IPFIX. *SIG COMM Comput. Commun. Rev., 44*(5), 20-26. doi:http://doi.org/10.1145/2677046.2677050

Howie, D. (2002). *Interpreting Probability : Controversies and Developments in the Early Twentieth Century*. Cambridge, UNITED KINGDOM: Cambridge University Press.

Ian Witten, Eibe Frank, Mark Hall, & Pal, C. (2016). *Data Mining: Pratical Machine Learning Tools and Techniques*: Todd Green.

Impact Cyber Trust. (2017). Researcher TOU for Quasi-Restricted. *Legal Framework.* Retrieved from https://www.impactcybertrust.org/tools_legal

Jackson, S. L. (2012). *Research methods and statistics : a critical thinking approach* (4th ed., [international ed.]. ed.). Singapore :: Wadsworth Cengage Learning.

Javed, M., & Paxson, V. (2013). Detecting stealthy, distributed SSH brute-forcingProceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 85-96). Berlin, Germany: ACM. Retrieved from https://dl.acm.org/citation.cfm?doid=2508859.2516719. doi:http://doi.org/10.1145/2508859.2516719

Jiang, H. (2017). A novel approach for forecasting global horizontal irradiance based on sparse quadratic RBF neural network. *Energy Conversion and Management, 152*, 266-280. doi:https://doi.org/10.1016/j.enconman.2017.09.043

Jiang, H., & Dong, Y. (2016). A nonlinear support vector machine model with hard penalty function based on glowworm swarm optimization for forecasting daily global solar radiation. *Energy Conversion and Management, 126*, 991-1002. doi:https://doi.org/10.1016/j.enconman.2016.08.069

Jin, F., Varadharajan, V., & Tupakula, U. (2017). An Eclat Algorithm Based Energy Detection for Cognitive Radio Networks2017 IEEE Trustcom/BigDataSE/ICESS (pp. 1096-1102). Sydney, NSW, Australia. Retrieved from https://ieeexplore.ieee.org/document/8029561/. doi:http://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.358

Jung, H., & Chung, K. (2015). Sequential pattern profiling based bio-detection for smart health service. *Cluster Computing, 18*(1), 209-219. doi:10.1007/s10586-014-0370-3

Jungsuk Song, Hiroki Takakura, & Okabe, Y. *Description of Kyoto University Benchmark Data*. Retrieved from http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf

Keith Stouffer, Victoria Pillitteri, Suzanne Lightman, Marshall Abrams, & Hahn, A. (2015). *Guide to Industrial Control Systems (ICS) Security*. Retrieved from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf

Kevric, J., Jukic, S., & Subasi, A. (2017). An effective combining classifier approach using tree algorithms for network intrusion detection. *Neural Computing and Applications, 28*(1), 1051-1058. doi:http://doi.org/10.1007/s00521-016-2418-1

Khalili, A., & Sami, A. (2015). SysDetect: A systematic approach to critical state determination for Industrial Intrusion Detection Systems using Apriori algorithm. *Journal of Process Control, 32*, 154-160. doi:https://doi.org/10.1016/j.jprocont.2015.04.005

Koch, R., Golling, M., & Rodosek, G. D. (2014). Behavior-based intrusion detection in encrypted environments. *IEEE Communications Magazine, 52*(7), 124-131. doi:http://doi.org/10.1109/MCOM.2014.6852093

Korczyński, M., & Duda, A. (2014). Markov chain fingerprinting to classify encrypted trafficIEEE INFOCOM 2014 - IEEE Conference on Computer Communications (pp. 781-789). Toronto, ON, Canada. Retrieved from https://ieeexplore.ieee.org/document/6848005/. doi:http://doi.org/10.1109/INFOCOM.2014.6848005

Kotiyal, B., Kumar, A., Pant, B., Goudar, R. H., Chauhan, S., & Junee, S. (2013). User behavior analysis in web log through comparative study of Eclat and Apriori2013 7th International Conference on Intelligent Systems and Control (ISCO) (pp. 421-426). Coimbatore, India. Retrieved from https://ieeexplore.ieee.org/document/6481192/. doi:http://doi.org/10.1109/ISCO.2013.6481192

Kudłacik, P., Porwik, P., & Wesołowski, T. (2016). Fuzzy approach for intrusion detection based on user's commands. *Soft Computing, 20*(7), 2705-2719. doi:https://doi.org/10.1007/s00500-015-1669-6

Kyoto University. (2015). *Traffic Data from Kyoto University's Honeypots*. Retrieved from: http://www.takakura.com/Kyoto_data/

Mackenzie, N., & Knipe, S. (2006). *Research dilemmas: Paradigms, methods and methodology* (Vol. 16).

Malley, B., Ramazzotti, D., & Wu, J. T.-y. (2016). Data Pre-processing *Secondary Analysis of Electronic Health Records* (pp. 115-141). Cham: Springer International Publishing.

Masduki, B. W., & Ramli, K. (2016). Improving intrusion detection system detection accuracy and reducing learning time by combining selected features selection and parameters optimization2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE) (pp. 397-402). Batu Ferringhi, Malaysia. Retrieved from https://ieeexplore.ieee.org/document/7893606/. doi:https://doi.org/10.1109/ICCSCE.2016.7893606

Mellor, D. H., & Koslow. (2004a). Probability : A Philosophical Introduction (pp. 23-25). London, UNITED KINGDOM: Routledge.

Mellor, D. H., & Koslow. (2004b). Probability : A Philosophical Introduction (pp. 50-52). London, UNITED KINGDOM: Routledge.

Michel Oosterhof. (2018a). Cowrie SSH/Telnet Honeypot. Retrieved from https://github.com/micheloosterhof/cowrie

Michel Oosterhof. (2018b). playlog. Retrieved from https://github.com/micheloosterhof/cowrie/blob/master/bin/playlog

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2014). Foundations of Machine Learning (pp. 1-10). Cambridge, UNITED STATES: MIT Press.

Moon, D., Pan, S. B., & Kim, I. (2016). Host-based intrusion detection system for secure human-centric computing. *The Journal of Supercomputing, 72*(7), 2520-2536. doi:https://doi.org/10.1007/s11227-015-1506-9

Moustaf, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective, 25*(1-3), 18-31. doi:https://doi.org/10.1080/19393555.2015.1125974

Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)2015 Military Communications and Information Systems Conference (MilCIS) (pp. 1-6). Canberra, AUS. Retrieved from https://ieeexplore.ieee.org/document/7348942/. doi:http://doi.org/10.1109/MilCIS.2015.7348942

Najafabadi, M. M., Khoshgoftaar, T. M., Calvert, C., & Kemp, C. (2015). Detection of SSH Brute Force Attacks Using Aggregated Netflow Data2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA) (pp. 283-288). Canbrra, AUS. Retrieved from https://ieeexplore.ieee.org/document/7424322/. doi:http://doi.org/10.1109/ICMLA.2015.20

Network Working Group. (2006a). *RFC 4252 - The Secure Shell (SSH) Authentication Protocol*. Retrieved from Network Working Group https://tools.ietf.org/html/rfc4252

Network Working Group. (2006b). *RFC 4253 - The Secure Shell (SSH) Transport Layer Protocol*. Retrieved from Network Working Group https://tools.ietf.org/html/rfc4253

Network Working Group. (2006c). *RFC 4254 - The Secure Shell (SSH) Connection Protocol*. Retrieved from Network Working Group https://tools.ietf.org/html/rfc4254

Pimenta Rodrigues, G., de Oliveira Albuquerque, R., Gomes de Deus, F., de Sousa Jr., R., de Oliveira Júnior, G., García Villalba, L., & Kim, T.-H. (2017). Cybersecurity and Network Forensics: Analysis of Malicious Traffic towards a Honeynet with Deep Packet Inspection. *Applied Sciences, 7*(10), 1082. doi:https://doi.org/10.3390/app7101082

R Studio. (2015). R Studio. Retrieved from https://www.rstudio.com/

Rabadia, P., Valli, C., Ibrahim, A., & Baig, Z. A. (2017). Analysis of attempted intrusions: intelligence gathered from SSH Honeypots15th Australian Digital Forensics Conference (pp. 26-35). Perth, W.A. Retrieved from http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1175&context=adf. doi:http://doi.org/10.4225/75/5a839e6d1d283

Ramsbrock, D., Berthier, R., & Cukier, M. (2007). Profiling Attacker Behavior Following SSH Compromises37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07) (pp. 119-124). Edinburgh, UK. Retrieved from https://ieeexplore.ieee.org/document/4272962/. doi:http://doi.org/10.1109/DSN.2007.76

SIGKDD. (1999). *KDD Cup 1999 Data*. Retrieved from: http://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data

Soheily-Khah, S., Marteau, P. F., & Béchet, N. (2018). Intrusion Detection in Network Systems Through Hybrid Supervised and Unsupervised Machine Learning Process: A Case Study on the ISCX Dataset2018 1st International Conference on Data Intelligence and Security (ICDIS) (pp. 219-226). South Padre Island, TX, USA. Retrieved from https://ieeexplore.ieee.org/document/8367767/. doi:http://doi.org/10.1109/ICDIS.2018.00043

SPSS. (2000). Step-by-step data mining guide. *CRISP-DM 1.0.* Retrieved from http://www.the-modeling-agency.com/crisp-dm.pdf

Sultana, N., Chilamkurti, N., Peng, W., & Alhadad, R. (2018). Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications, 11*(50), 1-9. doi:https://doi.org/10.1007/s12083-017-0630-0

SURF cert IDS. (2013). SURF cert IDS. Retrieved from http://ids.surfnet.nl/wiki/doku.php

Tapaswi, S., Mahboob, A., Shukla, A. S., Gupta, I., Verma, P., & Dhar, J. (2014). Markov Chain Based Roaming Schemes for Honeypots. *Wireless Personal Communications, 78*(2), 995-1010. doi:http://doi.org/10.1007/s11277-014-1797-9

Tatu Ylonen. (2017a). FTP – SECURE FILE TRANSFERS WITH SFTP. Retrieved from https://www.ssh.com/ssh/ftp/

Tatu Ylonen. (2017b). SFTP – SSH SECURE FILE TRANSFER PROTOCOL. Retrieved from https://www.ssh.com/ssh/sftp/

Tatu Ylonen. (2017c). SSH (SECURE SHELL). Retrieved from https://www.ssh.com/ssh/

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (pp. 1-6). Ottawa, ON, Canada. Retrieved from https://ieeexplore.ieee.org/document/5356528/. doi:http://doi.org/10.1109/CISDA.2009.5356528

The Department of the Prime Minister and Cabinet. (2016). *Australia's Cyber Security Strategy*. Retrieved from https://cybersecuritystrategy.pmc.gov.au/assets/img/PMC-Cyber-Strategy.pdf

The Honeynet Project. (2011). Dionaea - catches bugs. Retrieved from https://www.honeynet.org/project/Dionaea

Tsai, H. J., Chen, C. C., Peng, Y. C., Tsao, Y. H., Chiang, Y. N., Zhao, W. C., . . . Chen, T. F. (2017). A Flexible Wildcard-Pattern Matching Accelerator via Simultaneous Discrete Finite Automata. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25*(12), 3302-3316. doi:http://doi.org/10.1109/TVLSI.2017.2671408

Wang, X., Huang, X., Luo, Y., Pei, J., & Xu, M. (2018). Improving Workplace Hazard Identification Performance Using Data Mining. *Journal of Construction Engineering and Management, 144*(8). doi:https://doi.org/10.1061/(ASCE)CO.1943-7862.0001505

Wei, J., Liu, W., & Hu, X. (2016). Secure Data Sharing in Cloud Computing Using Revocable-Storage Identity-Based Encryption. *IEEE Transactions on Cloud Computing*, 1-1. doi:10.1109/TCC.2016.2545668

Williamson, K., & Johanson, G. (2013). *Research Methods: Information, Systems and Contexts*: Tilde University Press.

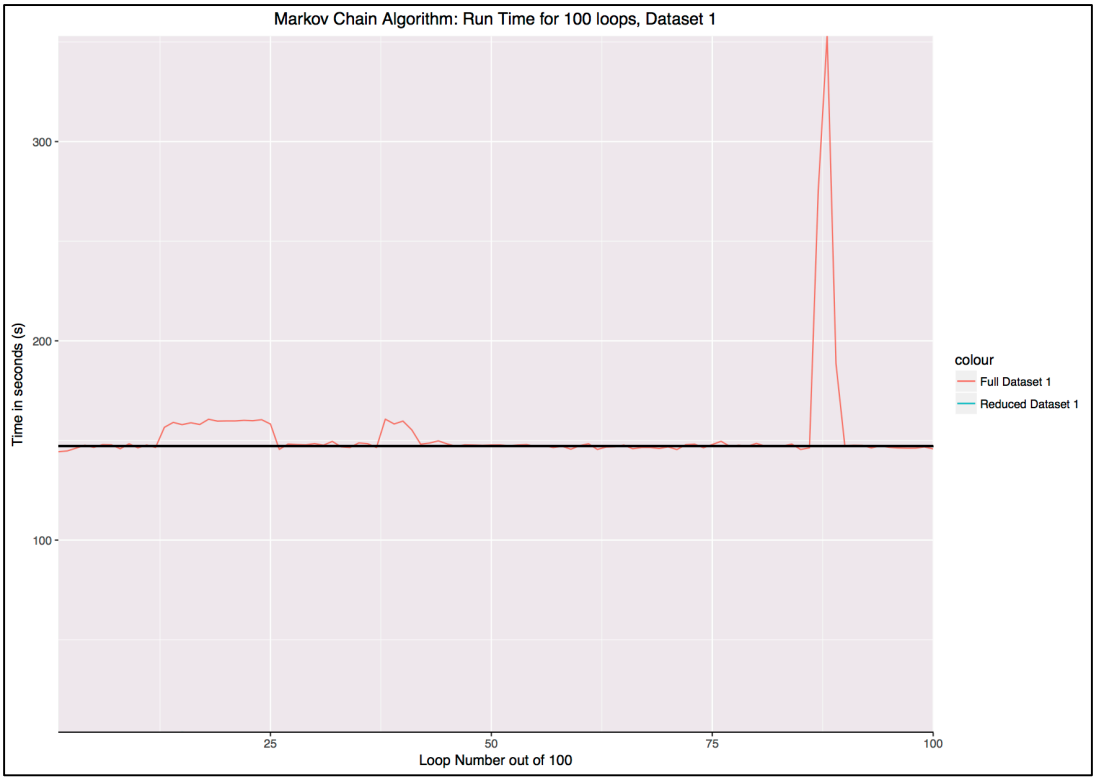
Xiao, L., Chen, Y., & Chang, C. K. (2014). Bayesian Model Averaging of Bayesian Network Classifiers for Intrusion Detection2014 IEEE 38th International Computer Software and Applications Conference Workshops (pp. 128-133). VÄSTERÅS, SWEDEN. Retrieved from https://ieeexplore.ieee.org/document/6903117/. doi:http://doi.org/10.1109/COMPSACW.2014.25

Appendix A

Package	*Version*	*Description*
arules	1.5-5	Association rule mining and frequent itemsets
arulessquences	0.2-1.9	Frequent sequence mining
datasets	3.4.1	R dataset package
dplyr	0.7.4	A grammar of data manipulation
gdata	2.18.0	Assortment of R programming tools for data manipulation
ggplot2	2.2.1	Represent data visually
gplot	3.0.1	Assortment of R programming tools for plotting data
graphics	3.4.1	The R graphics package
grDevices	3.4.1	The R graphics devices with support for colour and fonts
Gtools	3.5.0	Assortment of R programming tools
markovchain	0.6.9.8-1	Handle discrete time Markov chains
matrix	1.2-12	Spare and dense matrix classes and methods
methods	3.4.1	Formal methods and classes
microbenchmark	1.4-4	Accurate timing functions as well a visual representation of results
modeltools	0.2-21	Tools and classes for statistical models
multcomp	1.4-8	Simultaneous interface in general parametric models
mvtnorm	1.0-7	Multivariate normal and t distributions
naivebayes	0.9.2	Fit naïve bayes model to data for class predication
paralle	3.41	Support for parallel computation in R
parallelmap	1.3	Unified interface to parallelization Back-Ends
parallelml	1.2	A parallel-voting algorithm for many classifiers
plyr	1.8.4	Tool for splitting, applying and combining data
R6	2.2.2	Classes with reference semantics
readr	1.1.1	Read rectangular text data
reshape2	1.4.3	Flexibly reshape data: A reboot of the reshape package
rlang	0.6.1	Functions for base type and core R and “Tidyverse” feature
rpart	4.1-12	Recursive portioning and regression trees
rpart.plot	2.1.2	Rpart ‘rpart’ models: an enhanced version of ‘plot.rpart’
sandwich	2.4-0	Robust covariance matrix estimators
stats	3.4.1	The R stats package
stats4	3.4.1	Statistical functions using S4 classes
stringi	1.1.6	Character sting processing facilities
stringr	1.2.0	Simple, consistent wrappers for common string operations
tibble	1.4.2	Simple data frames
tictoc	1.0	Functions for timing R scripts, as well as implementations of Stack and List structure
utf8	1.1.3	Unicode text processing
zoo	1.8-1	S3 infrastructure for regular and irregular time series

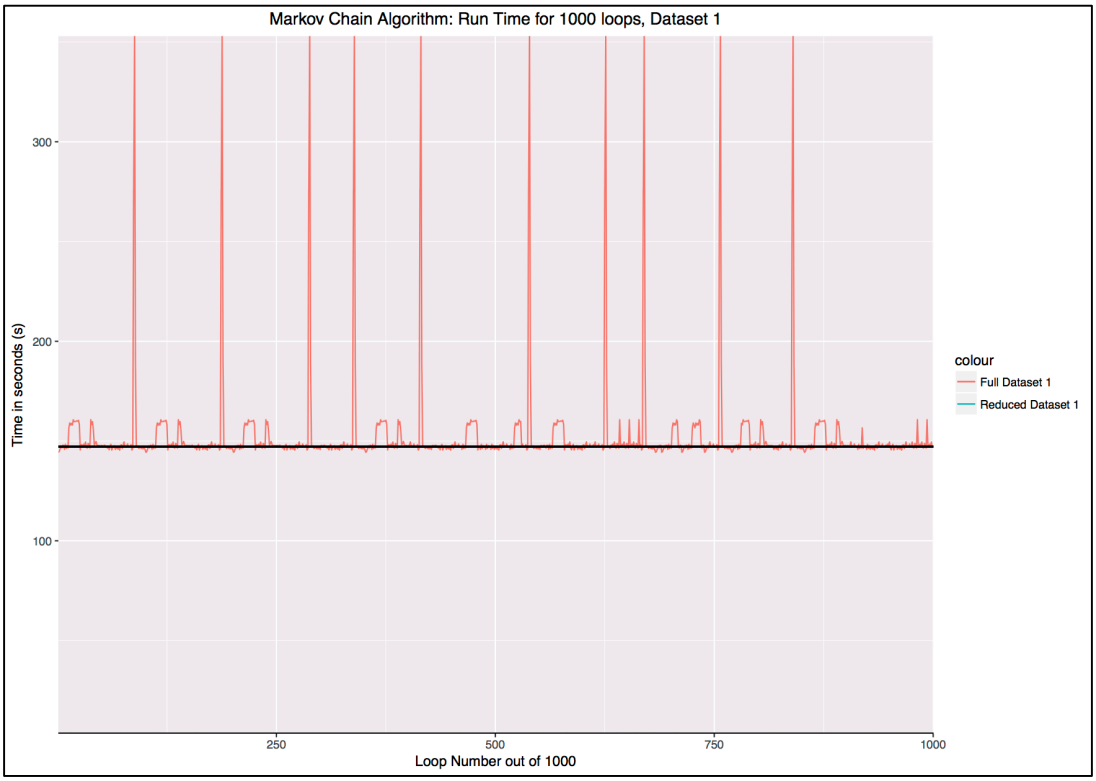
Appendix A, shows the R-Studio packages used within this study, for preliminary analysis and experimental phase

Appendix B



Appendix B, illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for the RD1 and FD1. The process was iterated 100 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed

Appendix C



Appendix C, illustrates the processing time in seconds(s) for the Markov chain algorithm to converge the transition matrices for the RD1 and FD1. The process was iterated 1000 times, with the black lines represent the mean processing time for the datasets after the outliers had been removed

Glossary

Adversary: An unauthorised user or entity on the network attempting to gain access to assets.

Activities: The interaction between two entities. For example: The interaction between an adversary and a system.

Association Rule Mining: The association class of machine learning algorithms are known as, association rule mining algorithms. The association rule mining algorithms addresses whether patterns of association are present between objects within a given dataset and is based on the frequentist theorem of probability. The Apriori and Equivalence Class Transformation (Eclat) algorithms are association rule mining algorithms.

Deep Pack Inspection: Further examination of packet level data is known as Deep Packet Inspection (DPI). DPI can be utilised for analysis in near real-time and off-line analysis.

Efficiency: One of the key attributes of an intrusion detection approaches. Data should be efficiently processed to allow for a timely detection of an adversary on the network prior to assets on a host or network being compromised.

Honeypot: Honeypots are an example of IDSs**.** Honeypots are decoy systems deployed to gather data on attempts made to gain access to a system. To an adversary, it is difficult to distinguish between a honeypot and a 'real' system, allowing the data gathered from honeypots to, accurately represent attempted intruder attacks to gain access to a system (Su, Chang, & Lin, 2009).

Intrusion Detection Approach: Intrusion detection approaches are deployed by Intrusion Detection Systems (IDSs), to monitor network traffic and trigger an alert or an appropriate response when unauthorised activities are detected on a network.

Intrusion Detection System (SURF cert IDS): These are systems deployed as part of a cyber security strategy. IDS are used to detected unauthorised activity on a system and execute relevant procedures when an intruder has been identified. An example of on IDS is a honeypot.

Intruder: An entity with the motive of gaining unauthorised access to a system.

Machine Learning: Machine learning is the application of learning algorithms to extract knowledge from data to determine patterns between data points and make predictions.

Precision: One of the key attributes of an intrusion detection approaches. Network traffic should be precisely classified, to identify a possible unauthorised attempt to gain access to assets on a host or network.

Probabilistic Classification: These algorithms are a sub-group, within the classification category based on the Bayesian theorem. The probabilistic classification algorithms explored are the Naïve Bayes algorithm and the Markov chain algorithm.

Sequence of commands: A chain of commands an adversary has used to attempt to gain unauthorised access to a system. For example, a chain of commands an adversary has use to interact with a medium interaction SSH honeypot.

Secure Shell (SSH): SSH is used for point to point communication over an insecure network, creating an encrypted tunnel for remote communications and access (Tatu Ylonen, 2017c). By default, SSH runs on the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) on port 22. SSH was developed in 1995 by Tatu Ylonen and was a response to a password sniffing attack against Helsinki University of Technology. This first version is referred to as SSH-1. The SSH service is predominately targeted by adversaries in order to attempt to gain unauthorised remote access to a system.