AN ENHANCEMENT OF CLASSIFICATION TECHNIQUE BASED ON ROUGH SET THEORY FOR INTRUSION DETECTION SYSTEM APPLICATION

NOOR SUHANA SULAIMAN

DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	: NOOR SUHANA BINTI SULAIMAN
Date of Birth	: 10 OCTOBER 1982
Title	: AN ENHANCEMENT OF CLASSIFICATION TECHNIQUE BASED ON ROUGH SET THEORY FOR INTRUSION DETECTION SYSTEM APPLICATION
Academic Session	: SEM 2, 2018/2019

I declare that this thesis is classified as:

☐ CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐ RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑ OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

_______________________	_______________________
(Student's Signature)	(Supervisor's Signature)
_______________________	_______________________
New IC/Passport Number Date:	Name of Supervisor Date:

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Doctor of Philosophy (Computer Science).

____

(Supervisor's Signature)

Full Name	: DR ROHANI BT ABU BAKAR
Position	: ASSOCIATE PROFESSOR
Date	:

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

____

(Student's Signature)

Full Name	: NOOR SUHANA BT SULAIMAN
ID Number	: PCC 11001
Date	:

AN ENHANCEMENT OF CLASSIFICATION TECHNIQUE BASED ON ROUGH SET THEORY FOR INTRUSION DETECTION SYSTEM APPLICATION

NOOR SUHANA SULAIMAN

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Doctor of Philosophy (Computer Science)

Faculty of Computer Systems & Software Engineering

UNIVERSITI MALAYSIA PAHANG

APRIL 2019

ACKNOWLEDGEMENTS

In the name of ALLAH, Most Merciful, Most Compassionate. Alhamdulillah, all praises to Allah for the strengths and blessings in completing this thesis. I would like to express the deepest sincere gratitude to my supervisor, Associate Professor Dr. Rohani Bt Abu Bakar, for her supervision and constant support. Her invaluable help of constructive comments and suggestions throughout the experimental and thesis works have contributed to the success of this research. My sincere appreciation also goes out to all my examiners; Dr. Mohamad Fadli Bin Zolkipli, Professor Dr. Kamal Zuhairi Bin Zamli, Professor Dr. Suhaidi Bin Hassan and Associate Professor Dr. Nur Izura Bt Udzir.

I am also obliged to express my greatest appreciation goes to my beloved parents; Mr. Sulaiman Bin Ramly and Mrs. Norihan Bt Abu Bakar, also goes out too Hj. Syed Noh Bin Syed Ali and Hjh C.A. Halimah Bt K.C. Ahammu. My deepest gratitude goes to my husband, Ansar Sadat bin M. Mohamed Kunju, childrens, Azeem, Ameen, Ameer, Sumayyah, Solehah and Syifaa, and also to my sisters, Noor Suhani Bt Sulaiman and Noor Suhaida Bt Sulaiman for their endless everlasting loves, sacrifices, prayers and encouragements throughout this journey.

I am very grateful towards all the staffs and lecturers of Faculty of Computer Systems & Software Engineering and Institute of Postgraduate Studies, Universiti Malaysia Pahang who have been directly or indirectly influential and supportive to this research. Sincere thanks to all my friends especially Sue, Kak Liza and others for their kindness and moral support during my study. Thanks for the friendship and memories. The sweet memory among us will never fade away. To those who indirectly contributed in this research, your kindness means a lot to me. Thank you very much.

ABSTRAK

Sistem Pengesan Pencerobohan mampu mengesan pencerobohan yang tidak dibenarkan ke dalam sistem dan rangkaian komputer dengan mencari punca serangan yang diketahui atau penyimpangan aktiviti normal. Walau bagaimanapun, prestasi ketepatan adalah salah satu isu dalam aplikasi Sistem Pengesan Pencerobohan. Sementara itu, pengkelasan adalah salah satu teknik dalam perlombongan data yang digunakan untuk meningkatkan prestasi Sistem Pengesan Pencerobohan. Untuk meningkatkan masalah prestasi klasifikasi, algoritma pemilihan ciri dan pembekasan adalah penting dalam memilih sifat yang berkaitan yang dapat meningkatkan prestasi klasifikasi. Algoritma pembekasan telah dicadangkan baru-baru ini akan tetapi, algoritma pembekasan tersebut hanya mampu mengendalikan atribut kategori dan tidak dapat menangani atribut berangka. Di dalam algoritma pembekasan, adalah sukar untuk menentukan bilangan selang dan lebar yang diperlukan. Oleh itu, untuk menangani dataset yang besar, teknik perlombongan data boleh diperbaiki dengan memperkenalkan algoritma yang berupaya untuk meningkatkan prestasi klasifikasi. Generasi peraturan dianggap sebagai proses penting dalam perlombongan data, malahan peraturan yang dihasilkan adalah dalam jumlah besar. Oleh itu, adalah mustahak untuk menentukan peraturan yang penting dan relevan untuk proses seterusnya. Oleh itu, tujuan kajian ini adalah untuk meningkatkan prestasi klasifikasi dari segi ketepatan, kadar pengesanan dan pengurangan kadar penggera positif palsu untuk aplikasi Sistem Pengesan Pencerobohan. Di dalam penyelidikan ini mencadangkan peningkatan algoritma pembekasan berdasarkan Pembekasan Tong dalam Teori Set Kasar untuk meningkatkan prestasi klasifikasi dan juga untuk meningkatkan strategi peraturan generasi dalam Teori Set Kasar dalam meningkatkan prestasi klasifikasi. Kedua-dua penambahbaikan ini dinilai dari segi ketepatan, penggera positif palsu dan kadar pengesanan terhadap data KDD Cup 99 dalam aplikasi Sistem Pengesan Pencerobohan. Beberapa algoritma pembekasan seperti Kesamaan Frekuensi Tong, Entropy / MDL, Naïve dan pembekasan yang dicadangkan telah dianalisis dan dibandingkan dalam kajian. Hasil eksperimen menunjukkan teknik yang dicadangkan mampu meningkatkan peratusan klasifikasi ketepatan sehingga 99.95%; dan bilangan tong yang minimum menentukan algoritma pembekasan yang baik. Impak dari kajian penyelidikanyang dicadangkan, peratusan kadar pengesanan serangan adalah meningkat dan kadar penggera positif palsu diminimumkan. Algoritma yang dicadangkan menghasilkan kompromi yang memuaskan antara bilangan tong dan juga ketepatan prestasi teknik klasifikasi.

ABSTRACT

An Intrusion Detection System (IDS) is capable to detect unauthorized intrusions into computer systems and networks by looking for signatures of known attacks or deviations of normal activity. However, accuracy performance is one of the issues in IDS application. Meanwhile, classification is one of techniques in data mining employed to increase IDS performance. In order to improve classification performance problem, feature selection and discretization algorithm are crucial in selecting relevant attributes that could improve classification performance. Discretization algorithms have been recently proposed; however, those algorithms of discretizer are only capable to handle categorical attributes and cannot deal with numerical attributes. In fact, it is difficult to determine the needed number of intervals and their width. Thus, to deal with huge dataset, data mining technique can be improved by introducing discretization algorithm to increase classification performance. The generation of rule is considered a crucial process in data mining and the generated rules are in a huge number. Therefore, it is dreadful to determine important and relevant rules for the next process. As a result, the aim of the study is to improve classification performance in terms of accuracy, detection rate and false positive alarm rate decreased for IDS application. Henceforth, to achieve the aim, current research work proposed an enhancement of discretization algorithm based on Binning Discretization in RST to improve classification performance and to enhance the strategy of generation rules in RST to improve classification performance. Both enhancements were evaluated in terms of accuracy, false positive alarm and detection rate against state-of-the-practice dataset (KDD Cup 99 dataset) in IDS application. Several discretization algorithms such Equal Frequency Binning, Entropy/MDL, Naïve and proposed discretization were analysed and compared in the study. Experimental results show the proposed technique increases accuracy classification percentage up to 99.95%; and the minimum number of bins determine good discretization algorithm. Consequently, attack detection rate increases and false positive alarm rate minimizes. In particular, the proposed algorithm obtains satisfactory compromise between the number of cuts and classification accuracy.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IoT	Internet of Things
UCL	University College London
ARPANET	Advanced Research Projects Agency Network
TCP	Transport Control Protocol
SSL	Secure Socket Layer
TLS	Transport Layer Security
HTTPS	Hyper Text Transfer Protocol Secure
DNS	Domain Name System
DDoS	Distributed Denial of Service
IT	Information Technology
WWW	World Wide Web
KDD	Knowledge Discovery and Data Mining
RST	Rough Set Theory
NIDS	Network intrusion detection systems
HIDS	Host Intrusion Detection System
DoS	Denial of Service
R2L	Remote to Local
U2R	Remote to User
GA	Genetic Algorithm
DT	Decision Tree
NNIV-RS	Neural Network with Indicator Variable Using Rough Set Theory for Attribute Reduction
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
TP	True Positive
FN	False Negative
TN	True Negative
FP	False Positive
FPR	False Positive Rate
FAR	False Caution Rate
FNR	False Negative Rate
TPR	True Positive Rate

TNR	True Negative Rate
MIT	Massachusetts Institute of Technology
DARPA	Defense Advanced Research Projects Agency
RIM	Rule Important Measure
MADAM ID	Mining Audit Data for Automated Model for Intrusion Detection

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Overview

Nowadays, Internet attacks are increasing rapidly. As a result, information security is a serious global concern among information technology users. In this chapter, Internet user growth and network security protection importance including the Intrusion Detection Systems (IDS) are discussed. They are elaborated in research background as research motivation of the study, problem statements, scopes and objectives and the contributions of the thesis.

1.2 Research Background

The introduction of Internet in July 1977 was a substantial scale exhibit of internetworking utilizing Advanced Research Projects Agency Network (ARPANET), parcel radio systems and satellite transmission (Hopgood, 2001). A message was sent from a van on San Francisco expressway by radio to ARPANET. From that point, the message was passed by satellite to Norway, by a land line to University College London (UCL), by satellite back to the United States of America (USA) and by ARPANET to Los Angeles (Hopgood, 2001). By 1985, there were 2000 hosts connected to Internet and by 1990, there were 2000 systems; and those were developed further to 94,000 systems by 1996. The latest data on Internet usage across the world (Internet Users, 2018) indicates that the world saw a billion online users on June of 2017. Based on Figure 1.1, Asia has gained the highest Internet user population with 2023 million users; meanwhile, the lowest Internet user is from Oceania/Australia with 28 million users. The increase in networked machines quantity has led to the expansion in illegitimate activities of internal and external attacks; for example, users gaining unprivileged access for individual gain (Saurabh & Sharma, 2012)

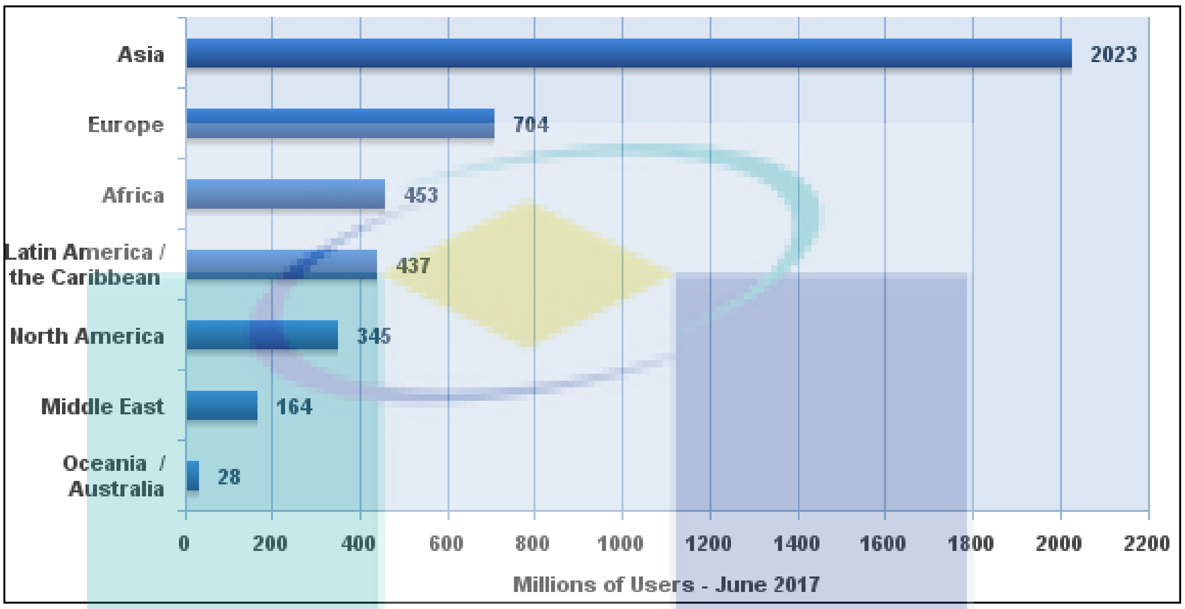

Figure 1.1 Internet Users in the World

Source: Internet Users (2018)

With the ever increasing users, object and device connected to each other through the Internet (i.e. Internet of Things (IoT)), have seen a steady increase of cloud systems to store data including credential data and paperless transaction as a result of online transactions. All these online technological dependencies expose users to high risk of public network / Internet breach of trusts such as compromised online credential data, unsafe communication between senders and receivers, and unauthorized access to communication session. In fact, given current rate of online technological growth, IoT based technologies are increasingly prone to vulnerabilities.

Different approaches are taken to deal with these vulnerabilities including the use of firewalls, adoption of cryptography techniques and introduction of secure protocols (named as https, SSL/TLS, IpSec, etc). Several protective techniques have been proposed and implemented to secure PC framework against cybercrime such as antivirus, firewall, encryption techniques and different protective measurements. Thus, Intrusion Detection System (IDS) is by no mean replacing all of these approaches but complementing them. Even with all the techniques carried out, it could not guarantee full protection of the system. Therefore, the protective domain needs more efficient mechanism like Intrusion Detection system (IDS) as the next track of defense (Dhakar & Tiwari, 2014). IDS is capable to detect unauthorized intrusions into computer systems and networks by looking for signatures of known attacks or deviations of normal activity. Referring to Figure 1.2, IDS can be considered as a burglar alarm for an office. When a user enters the office through an unauthorized access, the alarm will alert the user. Therefore, IDS is a detective control; its main function is to warn the user of any suspicious activity taking place.

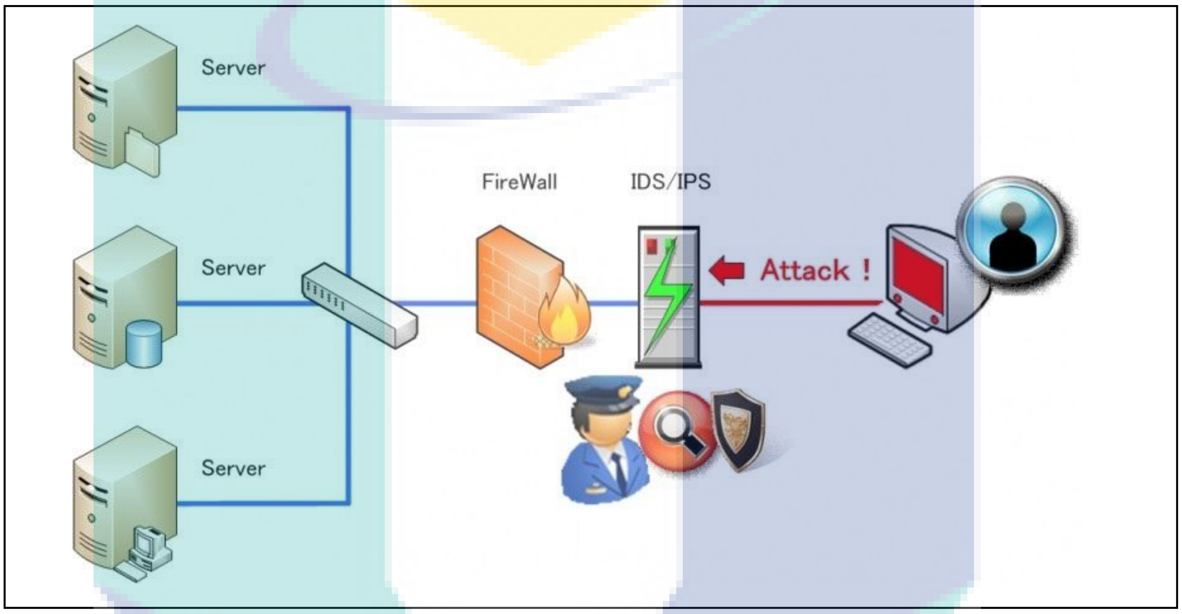

Figure 1.2 Intrusion Detection System Flows

1.3 Problem Statement

Data mining is one of techniques that can be used to address IDS performance, specifically in Preprocessing and Analysis phase. There are several data mining techniques such as Pattern Recognition, Clustering, Association and Classification that can be employed. To address problems in IDS, classification can be considered as an alternative to increase IDS application performance in terms of accuracy, complexity and precision. The problems in attack to classification are caused by low data quality, incorrect and missing values, attribute types, dominant classes presence and overfitting (retraining) as well as underfitting (weak model) problems (Vadim, 2018). Performance accuracy is one of data mining concerns as any algorithm can lost the property of accuracy and performance due to several factors. One of the factors is classification algorithm sensitivity to noisy data which causes the processing power of classification to slow down. Consequently, it decreases IDS performance. Hence, to improve classification performance problem, feature selection should be employed to select relevant attributes that improve classification performance (Ahmad et al., 2015).

IDS dataset normally contain numerical and categorical attributes in huge size. Nevertheless, many data mining algorithms deal with categorical attributes (Agrawal & Srikant, 1994; Dougherty et al., 1995; Liu, et al., 2002), as shown in Naïve Bayes (Yang & Webb, 2009) and Apriori's algorithms (Agrawal & Srikant, 1994). On the other hand, Rough Set Theory, which is a data mining technique, is able to deal with both attributes, (numerical and categorical attributes). However, another problem arises as how to determine the number of intervals and their width. Thus, to deal with huge dataset, data mining technique can be improved by introducing discretization algorithm to increase classification performance. Data mining algorithm performance should increase with the use of discretization algorithm (Garcia et al., 2013). Discretization is an essential part of data preprocessing to increase classifier performance by discretized the attributes into bin interval (Han & Kamber, 2006). The limitations of previous researches on discretization algorithm are elaborated in Chapter 2.

Meanwhile, the generation of rule is considered a crucial process in data mining. In the process, the generated rules are huge in number; hence, it is hard to ascertain which rules are important and relevant for the next process. Previous works on rule measures were identified such as Bruha (1997), Pang & Kumar (2000) and Hilderman & Hamilton (1999). Most of these works focus on how to measure relevant rules in rule evaluation process which do not contain knowledge from data domain. Thus, rule measures are insufficient to evaluate the rule whether the rule is relevant or otherwise into the certain domain (Li, 2007).

Given the aforementioned challenges, researchers are now focusing on integrating data mining classifier to enhance computational performance of IDS. As will be highlighted in Chapter 2, recent works include Support Vector Machine (Leandros et al., 2014), Fuzzy Logic (Sujendran & Arunachalam, 2015) Genetic Algorithm (Desale & Ade, Kaur et al., 2015; Liu et al., Wan et al., 2014) Decision Tree (Relan, 2015) and Rough Set Theory (Ashalata, 2018; Hui, 2016; Janmejay et al., Rampure & Tiwari, Anazida et al., 2015; Sadek et al., 2013). Although useful, the adoption of Rough Set Theory as part of data mining classifier into IDS has not been sufficiently explored.

Specific problems gathered from previous researches focus on classification issue in particular classification performance reflected from data quality and feature selection. Other specific problems highlighted include discretization issue in numerical and categorical attributes handling as well as the issue in determining the number of intervals. The last issue in this research is to determine which relevant rule from the large number of set rules has an effect on classification results. This research work presents an enhancement of classification technique based on Rough Set Theory in classifying attacks for Intrusion Detection System application to produce high accuracy and detection of performance. Furthermore, an enhancement algorithm of Frequency Binning was proposed for discretization process in Rough Set Theory. Finally, a new strategy was proposed and experimented to generate significant rules employed in the Rough Set Theory to improve the classification accuracy.

1.4 Research Objectives

The aim of thesis is to enhance the classification technique based on Rough Set Theory for Intrusion Detection System application particularly in terms of accuracy, detection rate and reduce the false positive alarm rate. In order to achieve this aim, the following objectives are outlined;

i) To propose discretization algorithm based on Binning discretization and strategy of rules in RST to improve classification performance.

ii) To integrate the proposed technique (i) to the RST classification to improve classification performance.

iii) To evaluate the proposed technique in terms of accuracy, false positive alarm and detection rate against state-of-the-practice dataset in IDS application.

1.5 Research Scope

This research was conducted based on RST in carrying out all objectives stated above. The intrusion detection dataset from “Knowledge Discovery and Data Mining

(KDD) 1999" is IDS benchmark dataset used as data problem for intrusion detection system. This dataset consists of 4,898,431 and 311,029 records in set of training and testing. Current research randomly used 20,996 records as training set and 8,999 records as testing set. The portion of records used as data in the study was determined by the limitation of the machine to process the data. The performance of validation part involves false alarm rate, attack detection rate and accuracy rate.

1.6 Research Significance

In this study, enhancements on the classification technique to IDS application with proposed Frequency Binning discretization and strategy of generating significant rules had impacted information and network security research domain to improve IDS classification performance. Those are in terms of accuracy, detection rate and false positive alarm rate degradation. Thus, online activities such as online shopping, virtual meeting, e-banking etcetera are safer with the employment of IDS application into network structure. IDS is better from previous applications as attack classification performances (accuracy, detection rate and lower false positive alarm rate) are increased.

1.7 Research Contribution

This research contributes to the body of knowledge in term of the enhancement of classification technique based on RST; as a result, contributes an efficient and enhanced intrusion detection performance in attack classification. The research also incorporates RST and enhancement of RST classification to improve IDS application performance. The main research contributions are as follow;

i) An enhancement of IDS application classification based on RST.
ii) An enhancement of discretization algorithm based on Frequency Binning for RST discretization process.
iii) An enhancement strategy to generate significant rules for RST rule process.

1.8 Thesis Organization

This thesis is organized into five chapters. Chapter 1 presents research background, research aim, objectives, scopes and research contributions of the study. Meanwhile, Chapter 2 reviews recent studies related to types of data mining and classification, Rough Set Theory, limitation of discretization and rules, IDS dataset and various steps which make up the new method, followed by Chapter 3 which describes in details the proposed classification enhancement to improve IDS application. Then, Chapter 4 discusses simulation results executed to assess the validity of proposed algorithm and strategy. Finally, Chapter 5 concludes the research and offers recommendations for the improvements to this work.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

This chapter presents a comprehensive literature review including data mining and classification techniques, limitation of discretization and rules strategy from previous works, IDS models and benchmark data. Apart from that, previous related studies and the technique used in IDS application classification are reviewed. Additionally, Rough Set Theory is introduced as the base for the proposed enhancement.

2.2 Techniques of Data Mining

Techniques of data mining deal with knowledge or interesting patterns search from the massive data. Data mining turns large data into knowledge; then analyzes and generates the knowledge into information which is an important step in knowledge discovery process (Lakshmi & Raghunandhan, 2011). Data mining has been applied widely in the areas of IDS application, education, medical diagnosis, fraud detection and banking (Singh et al., 2013).

Knowledge Discovery in Databases (KDD) is a domain which encompasses theory, method and technique to make sense of data and extract useful knowledge from dataset. A set of standard is employed including selection, preprocessing, transformation, data mining and interpretation or evaluation to generate the knowledge, as depicted in Figure 2.1. In KDD data analyzing, data mining is the most important step in data processing (Kavakiotis et al., 2017).

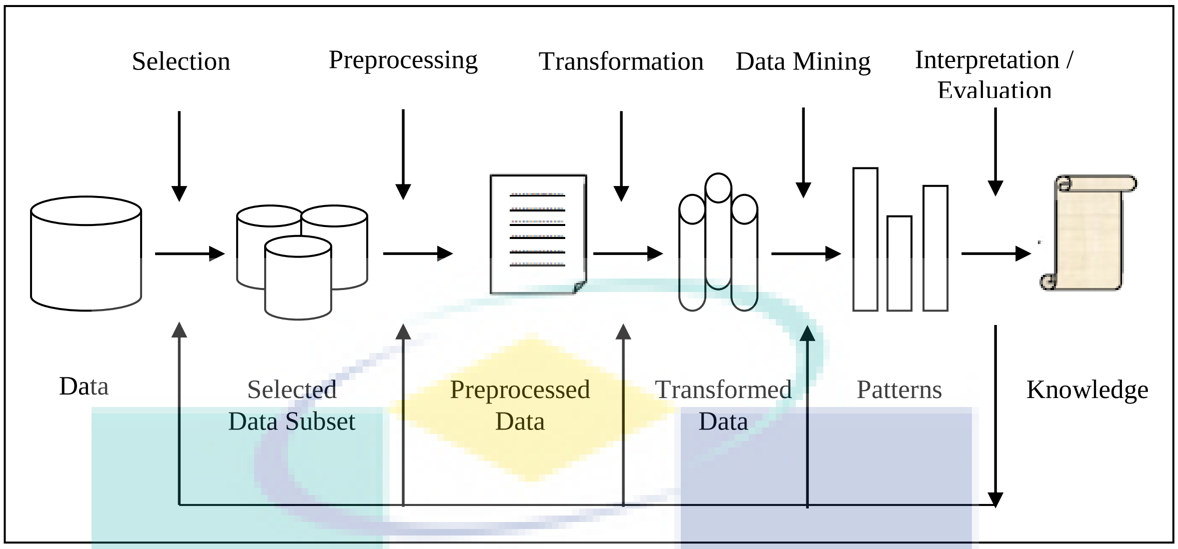

Figure 2.1 Standard Steps of Process in Knowledge Discovery Database (KDD)
Source: Kavakiotis et al. (2017)

Three techniques of data mining applied to generate knowledge discovery are as follow; (Gera & Goel, 2015)

i) Classification; is a process of classifying data based on class label, by predicting group membership for instances of data;

ii) Clustering; is a process of partitioning dataset or object into meaningful subclasses (clusters), by separating set of unlabeled data into hidden data and natural discrete set;

iii) Regression; is the process of finding function or model to distinguish data into continuous real value instead of using classes. This technique estimates value by comparing between already known value and predicted value.

The classification techniques involved in this study is further explained in the next section.

2.3 Classification

Classification technique is a data analysis task to extract model consisting of classes of data or to predict future trends of data. Prediction is a process to predict data that can belong to which class. Normally, attributes are categorized into two types; output or dependent attribute, and input or independent attribute. Samples of classification technique available in the data mining are decision tree induction, rule based classification, backpropagation classification and lazy learners. For instance, decision tree induction is used to predict output for continuous attribute. It is possible to have error in class prediction for decision tree due to bias variance and overfit model. On the other hand, rule based classification is represented by *if- then* rule set in the developed model or algorithm. Backpropagation classification is a learning algorithm of neural network, which performs data processing iteratively. In this technique, data processing is learned by comparing the obtained result with earlier given target value. Lazy learner algorithm supports incremental learning process for the given tuple of training. The algorithm simply stores it and waits until a tuple of test is given. The examples of lazy learner algorithm are cases based on reasoning and K-Nearest Neighbor (Gera & Goel, 2015).

The categories of classification problem solving called supervised learning and unsupervised learning are explained in the following sections.

2.3.1 Supervised Learning

In supervised learning, firstly a classifier model needs to be developed. Then, a qualifier makes learning stage to check the expected classification result, which depends on the type of artificial intelligence approach involved. However, if the classification result is inaccurate, additional training of classifier is needed. The stage continues until it reaches the level of quality desired or the algorithm works incorrectly or the data does not have an identified structure (Vadim, 2018). Supervised learning trains the algorithm with the labelled samples. Then, the trained algorithm can predict unlabelled samples which are similar to the samples. The process includes knowledge extraction, prediction and compression tasks (Hamid et al., 2016).

One related classifier of supervised learning is K-Means. K-Means algorithm is a simple clustering used in result testing. K-Means algorithm provides an indication to the problem solved or the opposite. However, there is no guarantee that cluster made by K-means algorithm is correct. Another classifier example is One Class Support Vector Machines. Its classifier uses binary classification with fast execution. The classifier pre-processes the data to check for any abnormal behavior condition before passing the condition to other algorithms to be processed into training and testing set (Repalle & Kolluru, 2017).

2.3.2 Unsupervised Learning

Unsupervised learning consists of tasks combination and descriptive models. Unsupervised classification can be called as clustering which is also known as data analysis. Clustering process is to separate unlabeled dataset into finite data, hidden data structure and natural discrete set. This technique does not provide characterization accuracy of unobserved samples generated by the same probability distribution (Gera & Goel, 2015). Meanwhile, the advantage of unsupervised learning is that it enables to solve problem occurs without any prior knowledge on the analyzed data (Vadim, 2018). Unsupervised learning does not need any sample of training set. It uses statistical approach of density estimation to find clusters of hidden data or to group similar data. The process includes pattern recognition and outlier detection tasks (Hamid et al., 2016).

Support Vector Machines (SVM) is among popular algorithms in data mining implementation. SVM is able to classify linear and non-linear processes and promises accurate result as classification output. K-Nearest Neighbors is also one of the promising algorithms in the classification. The classification happens on the basis of different neighbors instead of trying to make classifier seems to fit better the featured data. Nonetheless, there are other related machine learning algorithms such as Decision Tree and Bayesian algorithms. Both classifiers seem less promising for detecting problems. The differences between normal and abnormal data are minimal and it seems that these algorithms produce more errors due to bias variance value and overfit model. Decision Tree and Bayesian algorithms are still used in the implementation of IDS to analyse and classify the correct or incorrect data (Repalle & Kolluru, 2017).

2.4 Rough Set Theory

Rough Set Theory (RST) algorithm was proposed by Polish philosopher, Professor Zdzisław Pawlak (1982) for adapting unclear and imprecise concept. RST is among the highest growing soft computing techniques in identifying and recognizing data common patterns, including the uncertain and incompleted data cases (Pawlak, 1997). RST analyzes and partitions the knowledge based on the perspective of universe; whereby, formal express classification of equivalence relation is used. RST has been widely applied in knowledge acquisition (Ashalata, 2018; Rampure & Tiwari, 2015; Guoyong, 2001), rule extraction (Sadek et al., 2013; Li, 2007; Li & Cercone, 2005), machine learning (Atilla & Hamit, 2016), decision analysis (Shen et al., 2012), pattern recognition (Shen et al., 2012) (Kumar & Dhawan, 2012), data mining (Kumar & Yadav, Anazida et al., 2015; Sengupta et al., 2013) and other fields (Koller & Sahami, 1996).

Figure 2.2 shows basic RST classification model (Pawlak, 1991). Data went through preprocessing or denoising step to eliminate missing and irrelevant features. Then, the data was discretized to segregate continuous to interval attributes. Discretization process is important to gain accurate classification result. After the discretization step, data was divided into two portions which are training and testing sets. Training dataset went through reduct process to remove unnecessary attributes from dataset to generate precise classification output. Finally, those generated rules were used by testing the dataset to classify them.

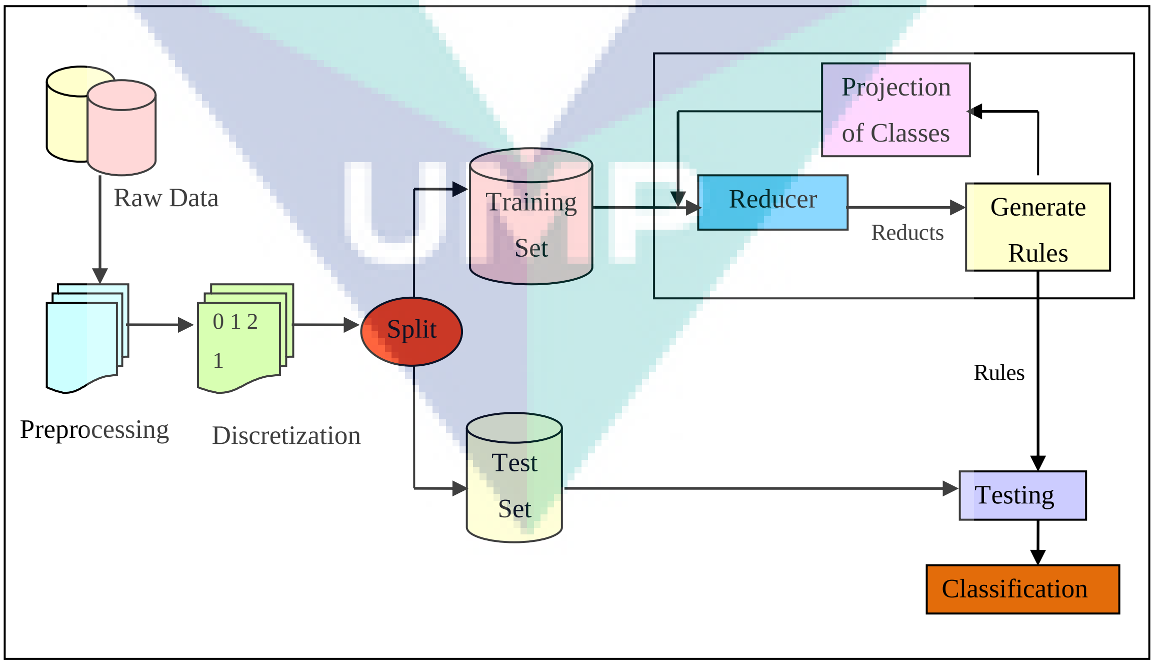

Figure 2.2 Standard Rough Set Theory Classification Model
Source: Zdzislaw Pawlak (1991)

In Rough Set approach, ten concepts must be defined and the details are as in the following;

i) Knowledge representation to the decision table / information system

The knowledge base in processing of RST is written as in Table consisting conditional and decision attributes. The Table serves to store all data; whereby, columns represent attributes, rows represent objects and cells contain attribute values for object and attribute. It is worth highlighting that the Table represents the knowledge in relation to the *If-Then* principles.

ii) Indiscernibility Relations

The indiscernibility relation is implied due to lack of knowledge from its unability to discern several objects with regard the information available. Let B be non-empty subset of set A of all attributes and $\rho: U \times A \rightarrow V$ is information function. The indiscernibility relation $IND(B)$ is relation on U defined for $x,y \in U$ as;

$$(x, y) \in IND(B) \text{ if and only if } \rho(x,a) = \rho(y,a) \text{ for all } a \in B \qquad 2.1$$

The indiscernibility relation $IND(B)$ is also called as equivalence relation. Corresponding partition on U is denoted by B (Jerzy, 2000).

iii) Equivalence Class

Basic RST model tells whether one object x belongs to object set X or not. It depends on eloquent understanding whether the equivalence class containing object x is contained by set X or not. Equivalence Relation $R(A)$: A is a subset of AT, attributes. The indiscernibility generated from a set of equivalence relations is formed by the intersection of a set of equivalence class from equivalence relation (Pawlak, 1991).

iv) Set Approximation

Space of approximation required equivalence relation set additionally contains formal method that maps any equivalence relation subset to equivalent relation on universe. The idea of space approximation consequence analytical on approximation classification in handling object classification belongs to a certain universe under insufficient information (Pawlak, 1991). Let be an information system *A* = *(U, A)*, let $X \subseteq U$ be set of objects and $B \subseteq A$ be selected set of attributes *B* lower approximation $\underline{B}X$ is;

$$\underline{B}X = \{ x \in U : [x]B \subseteq X \} \qquad 2.2$$

Lower approximation is a set of objects classified into decision *X*. Objects have equivalence class contained in set *X*. All objects can be discerned from outside of set $\underline{B}X$ (Guoyong, 2001). Description of object domain which contains all objects that may belong to a concept is an upper approximation.

$$\bar{B}X = \{x \in U : [x]_B \cap X \neq 0\} \qquad 2.3$$

v) Discernibility Matrix

Discernibility matrix is generated by listing all objects on row and column axis. In matrix entry, the differences are stated between objects of row and column axis (Pawlak, 1991).

vi) Discernibility Function

When the discernibility matrix is generated, discernibility function can be defined. This is a concise notation of how each object in the data can be discerned from the others (Pawlak, 1991). Let be an information system, *A*= *(U, A)*, the discernibility function of *A* over $B \subseteq A$ is;

$$f[B] = \wedge \vee m[B] \ (Ei, Ej) \ i,j \ \hat{I} \ \{1, \ldots n\} \qquad 2.4$$

A set B is a subset of A if and only if all elements in *B* are also in *A*. This is denoted by $B \subseteq A$.

vii) Discretization

This process divides the range values division into *k* intervals with equal width. For example, the values are divided into *k* bins based on equal frequency in Frequency Binning algorithm; consequently, each bin has the same instance number.

viii) Reduct

The aim of reduct is to discard unnecessary attributes from dataset, which become obstacle to generate precise classification output. Reduct contain core attributes and weak relevant attribute. Each reduct is satisfactory to determine concept in dataset. Refer to set of reducts, some attribute selection criteria reducts contain minimal attributes set (Nguyen, 1998).

ix) Rule Generation

Decision rule is presented as an *If-Then* formula $Dx: ANT \rightarrow CONS$, where $ANT = \wedge (ci, vi)$, $cons = (d, v)$, $ci \in P$ is a condition attribute, decision attribute is *d*. *ANT* is called antecedent which is the condition part, logic *AND* the attribute value pairs; and *CONS* is called consequent which is the decision part. The attribute value pair (ci, vi) is one of elementary conditions, which means $ci\ (Dx) = vi$; whereas (d, v) is elementary decision (Guoyong, 2001). The generation of rules, which plays a crucial role in predicting the output, is a unique feature of RST method. Rosetta has listed the rules and some statistics for the rules, i.e., support, accuracy, coverage, stability and length. The meaning of rule statistics is stated in the following (Bose, 2006). It can be generated by ROSETTA tool (is further explained in the next section) and was used in the research experiments;

i) Rule LHS support: training data record which fully exhibits property, which is described by IF condition.

ii) Rule RHS support: training data record which fully exhibits the property, which is described by THEN condition.

iii) Rule RHS accuracy: RHS support number is divided by number of LHS support.

iv) Rule LHS coverage: record fraction which satisfies IF conditions of the rule. It is obtained by dividing the support of rule with total number of records in the training sample.

v) Rule RHS coverage: training record fraction which satisfies THEN conditions. It is obtained by dividing the support of rule with number of records in training that satisfies THEN condition.

vi) Rule length: conditional elements numbers in IF part.

x) RST Classification

Learning function process which allocates object data to subset of class set is called classification. Hence, classifier is trained with training objects label set to specify each class by generating rule set. When a classifier is given another case, the rule set is examined to discover relevant rule. If more than one rule match, more than one possible outcomes are indicated. A voting procedure is executed over the rule match to resolve conflicts and to rank the anticipated results. Numerous standard algorithms for classification were used in RST such as First Rule, Highest Accuracy, Simple Voting, Quadratic Voting, Exponential Voting and Weight of Evidence (Pawlak, 1991).

2.5 Limitation of Exising Discretization and Rule Generation Research

Discretization can be performed as supervised or unsupervised method. In unsupervised method, the discretization can take class information or otherwise into account. In supervised method, discretization can take advantage of class label known in training data, especially if learning algorithm is involved in model building. In Rough Set Theory, several standard discretization algorithms named as Equal Frequency Binning, Boolean Reasoning, Naïve Bayes and Entropy to name a few are identified. This section elaborates Table 2.1 relating to available discretization algorithm in several years. The algorithms of data mining in Table 2.1 are only capable to handle categorical attribute and cannot deal with numerical attribute (Liu, et al., 2002; Dougherty et al.,1995; Agrawal & Srikant, 1994). Examples can be found in Naïve Bayes (Yang & Webb, 2009) and Apriori (Agrawal & Srikant, 1994) due to discretizer incapability to process numeric attributes. Meanwhile, the dataset consist of numerical and categorical attributes. In contrast, some other data mining algorithms like Boolean Reasoning, Equal Frequency Binning and Entropy can handle numerical attributes. For this reason, discretization algorithm (Yang & Webb, 2009; Kurgan & Cios, 2004;) is considered vital to data mining domain and process. Hence, data mining efficiency often improves with the benefit of discretization algorithm (Garcia et al., 2013).

The discretization technique of Equal Width Binning is applied to the unsupervised discretization as in Equal Frequency Binning (Kerber & Chimerge, 1992; Catlett, 1991). Basically, the process is the range value division into *k* intervals of equal width. Meanwhile, Equal Frequency Binning divides the values into *k* bins of equal frequency; as a result each bin has the same instance number. Since the class label is not involved, the discretization process possibly loses the classification information due to different classes of instances that can be easily grouped together. The result is not convincing as values of continuous distribution are ununiformed. Moreover, it is vulnerable to outliers because they affect the range values significantly (Catlett, 1991). Boolean Reasoning is another discretization algorithm which takes the value over underlying set using the Associated Boolean Algebra. The adoption of calculation of propositional rules to Boolean problem can produce incorrect result (Allen, 1990). Boolean Reasoning is a supervised technique which discretizes attributes and generates smaller interval value.

Discretization is also considered as essential in data preprocessing to increase result quality obtained by data mining algorithm (Han & Kamber, 2006). In decision tree, this technique typically divides the variable values into two parts based on appropriate threshold values. Two key problems in association with discretization are;

i) How to select the number of intervals or bins
ii) How to determine their widths.

Mitra et al., (2017) employed Modified Minimum Information Loss (MIL) discretization algorithm. The aim of MIL is to increase classification accuracy and to minimize information losses while discretizing continuous attributes. The proposed algorithm of classification percentage performs higher result compared to the original Modified Minimum Information Loss and Minimum Description Length Principle because of the computed standard deviation values used in their proposed work. The problem of MIL is that the user needs to set the initial number of subintervals in each continuous attribute. Their future works involve deciding the number of distinct subintervals for each continuous attribute.

Liyana et al., (2011) utilized nutrition dataset in their research. They proposed a model of classification using Boolean Reasoning and Entropy discretization algorithm. The aim of their research was to compare both discretization algorithms in order to determine suitable algorithm for nutrition dataset. The experimental outputs depicted that Boolean Reasoning worked better than Entropy which produced higher accuracy classification with 89.66%. The reason for Boolean Reasoning to perform well for the dataset because this algorithm produces less number of intervals. Even though, Boolean Reasoning algorithm performance is better compared to Entropy, shorter rules generated in Boolean Reasoning may contribute to knowledge losses.

Table 2.1 Related Works on Discretization Algorithms

Author	Year	Algorithm	Findings
Mitra, Sundereisan, & Sarkar	2017	Modified Minimum Information Loss	Only one discrete value is generated if density example occurs only at the last sub interval.
Nor, Azuraliza, Zulaiha	2011	Boolean Reasoning & Entropy	Boolean Resoning produce better classification percentage compared to Entropy. Less interval number guaranteed better accuracy, however if interval number are too small, certain information will be loss.
Yang & Webb	2009	Naïve Bayes	Algorithm can only handle categorical attributes
Han & Kamber	2006	Decision Tree	Divide variable values into two parts based on the appropriate threshold values

Table 2.1 Continued

Author	Year	Algorithm	Findings
Agrawal & Srikantt	1994	Apriori	Algorithm can only handle categorical attributes.
Catlett	1991	Equal Frequency Binning	Discretization process possibly loses the classification information, due to different classes of instances can be easily grouped together.
Allen	1990	Boolean Reasoning	Adoption of calculation of propositional rules to Boolean problem cans consequent to incorrect result.

As mentioned previously, the second crucial problem in data mining is the process of generating rules. Some rules are used in prediction process. A problem in the generation of rules in RST is the number of rules generated from data mining algorithm; whereby large dataset produces large rule set. Therefore, it is difficult to determine which rules are important and relevant. Relevant rule measures and rule quality measures from statistics and information theory areas were studied (Pang & Kumar, 2000; Hilderman & Hamilton, 1999; Bruha, 1997). Rule objective measurement is to measure the rules without any predefined opinions. Most previous studies (Azevedo & Jorge, 2007; Carvalho et al., 2005) focus on objective measures in dealing with rule evaluation, which do not contain knowledge from data domain. Thus, the objective measures are insufficient to evaluate the relevancy of a rule to certain domain (Li, 2007).

Table 2.2 shows several related works from year 2010 to 2015. Simranjit & Ruhi (2015) employed Apriori algorithm and support function in generating rule set, and applied Market Basket Analysis to identify a set of products frequently purchased by customers. The authors reported utilizing support function, whereby new Aprori algorithm was implemented by adding profit weight factor to particular item purchased, did not produce much profit. Meanwhile, Arti et al. (2013) applied Market Basket Analysis for decision making and understanding customer’s behavior. They utilized Apriori algorithm and support as well as confidence functions to generate rules set. Magdalene (2013) employed Apriori algorithm to extract student’s performance pattern to analyze and improve their performance for the purpose of class placement or to determine other privileges. Du & Gao (2010) stated that Apriori algorithm causes bottleneck problem in generating item set of rules and is time consuming to complete the process. In Goswami et al. (2010) study, several algorithms were proposed to generate frequent patterns in rules set namely Apriori algorithm, record filter, intersection and proposed algorithms. The proposed algorithm based on Apriori algorithm resulted better classification percentage compared to the aforementioned algorithms.

Table 2.2 Related Works on Generating Rules

Author	Year	Description	Algorithm
Simranjit Kaur & Ruhi Bagga	2015	Use Market Basket Analysis to identify a set of products which customers frequently purchase together.	• Use Apriori Algorithm. • Use Support function.
Arti Rathore, Ajaysingh Dhabariya & Chintan Thacker	2013	Extract interesting correlations, rules, frequent patterns and associations among sets of items in the transactional databases.	• Use Apriori Algorithm. • Use Support and Confidence functions.
Magdalene Delighta Angeline	2013	Extracted rules helps to predict the performance of the students and it identify the average, below average and good students.	• Use Apriori Algorithm. • Use minimum support and confidence function.
Du Ping & Gao Yongping	2010	Improve Apriori Algorithm	• Construct user interest itemsets, reduce unnecessary itemsets. • Use support function.
Goswami, Chaturvedi Anshu & Raghuvanshi	2010	A number of algorithms has been proposed to determine frequent pattern. Apriori algorithm is the first algorithm proposed in this field.	• Record filter, Intersection and Proposed Algorithm.

2.6 Related Research on Rough Set in IDS Application

Many data mining classifiers are proposed in IDS such as Support Vector Machine (Leandros et al., 2014), Fuzzy Logic (Sujendran & Arunachalam, 2015) Genetic Algorithm (Desale & Ade, Kaur et al., 2015; Liu et al., Wan et al., 2014) Decision Tree (Relan, 2015) and Rough Set Theory (Ashalata, 2018; Hui, 2016; Janmejay et al., Rampure & Tiwari, Anazida et al., 2015; Sadek et al., 2013).

Shen et al. (2012) proposed Improved Artificial Immune System (AIS) for Intrusion Detection System based on Rough Set. RST is adapted to present work to reduce complexity and maintain intrusion detection performance. The dataset used is KDD Cup 99 and a number of testings were conducted to evaluate the presented work. The results depicted the improvement of accuracy and detection rate at 98.25%. Supposed that maybe some adaptive mechanisms will be introduced to AIS, the intrusion detector can be updated to adapt to the changes in all situations of network condition.

On the other hand, Jaisankar et al. (2012) proposed an Intelligent IDS using Fuzzy Rough Set Based C4.5 classification algorithm in order to increase intrusion detection accuracy. Fuzzy Rough Set based Outlier Detection algorithm functions to preprocess the input data. KDD Cup 99 dataset was used in experiment simulations. The results show that false alarm reduced and accuracy detection rate increased when the proposed model was tested.

Zhang et al. (2012) employed Rough Set Theory (RST) and Support Vector Machine (SVM) in detecting intrusions. In the beginning, captured network packets undergo preprocessing step to reduce the dimension. RST selects features to be sent to SVM model, to learn and to test respectively. The experiment outcomes show that RST and SVM decreased false positive rates and increased detection accuracy with the reduct of featured numbers to 29 from 41. For future research, number of testing data should be increased to find accuracy variation.

Furthermore, Sadek et al. (2013) proposed Neural Network with Indicator Variable using Rough Set (NNIV-RS) algorithm to reduce the required time consumption of computer resources, as for example from memory and Central Processing Unit (CPU) in attack detection. Neural Network only classifies numerical data in training and testing. Thus, feature selection process is executed as in Figure 2.3 by employing Johnson Algorithm of Rough Set Theory in attributes reduct.

```
Convert categorical features to numeric values.
i)      All features in numeric values
            i.  Set protocol label: (TCP=3, UDP=7, ICMP=9).
ii)     Convert attack type to category:
            i.  0 = Normal, 1 = DoS (Denial of service), 2 = probe, 3 = R2L,
                4 = U2R.
```

Figure 2.3 Feature Selection Process Using Rough Set Theory
Source: Sadek et al. (2013)

Then, Neural Network classifies the network packet involving preprocessing step to process an input to desired output. Results from proposed work indicate that detection rate in indicated at 96.7% with false alarm rate at 3%.

On the other hand, Sengupta et al. (2013) carried out a research on online IDS using modified Q-learning algorithm with Rough Set Theory as in Figure 2.4. The algorithm includes the selected features and calculates classification accuracy to reduce computational cost in order to improve the output result. Different cut for different attribute yields the best accuracy classification result.

```
1   Start
2       Find the attributes appearing most frequently (at least twice).
3       Apply ''AND'' operation on the terms having such attributes and ''OR''
        operation on the rest.
4       Apply the connective ''AND'' between the ''OR'' terms and the term if
        consisting of such attribute then eliminate.
5       Combine the terms, obtained from (ii) and (iii) using ''AND'' operation.
6   End
```

Figure 2.4: Expansion Law Algorithm
Source: Sengupta et al. (2013)

The proposed method was tested against different related dataset. As a result, the observed result achieved higher accuracy classification compared to other same domain of existing classifiers. Figure 2.4 shows classification algorithm based on RST expansion law for discernibility function. It was observed that for discretization of continuous attribute, if same cut is applied to all attributes, classification accuracy varies widely even for two consecutive values of cut. However, the combination of different cuts for different attributes yields best result of classification accuracy. Future research suggested in this paper focuses on novel class detection due to changing data characteristic such as space and time. The classifier needs to be designed and fused with different approaches to increase classification accuracy in IDS.

Hamid et al. (2015) state that in RST does not guarantee optimal reduct of KDD Cup 99 subset due to the overlapping between lower and upper approximation in each class and reduct. This research proposed to enhance class reduct by overcoming overlapping rough set problem via union and voting attributes addition to classes of dataset as enhanced reduct activity. Reducts were evaluated by using different classification algorithms resulting in the achievement of high and comparable accuracy rates in the same dataset. Figure 2.5 shows pseudo code of proposed approach by Hamid et al. (2015).

```
1    Start
2        Apply RST algorithm to each class of KDD Cup 1999
3        Define normal attributers set of each class
4        Find union attributes set of all classes normal attributes
5        Apply voting technique to find voting attributes set
6            For all normal attributes of all class:
7             Find the repetition of each attributes
8            If the number of attribute repetition is $\geq 3$ adds the attribute to the
             voting attribute set
9             Define the voting attribute set
10        Apply classification algorithm to evaluate normal, union and voting attributes
          sets
11        Set with high accuracy = final attributes set
12   End
```

Figure 2.5 Enhancing RST Attributes Selection Using KDD Cup 99 Algorithm
Source: Hamid et al. (2015)

Supposed the proposed study should apply the same approach to several reducts of same algorithm for the dataset and with the different datasets.

Janmejay et al. (2015) proposed basic RST in selecting IDS features. Johnson reduct algorithm was used to find one reduct, as the output is a set of reducts and a set of "*If-Then-Else*" rule. Following is the process of feature selection using RST as proposed by Janmejay et al. (2015).

```
1   Start
2           Choose approximate dataset for intrusion detection.
3           Split the dataset into two dataset; training dataset and testing dataset.
4           Convert into suitable file format Refer to the valid data extension
            that simulation the tool supports.
5           Choose the feature selection model
6           Choose the feature selection techniques Refer to the feature selection
            model.
7           Evaluate the feature selection technique with learning algorithm that
            is classifier.
8   End
```

Figure 2.6 Feature Selection Process Using Rough Set Theory

Source: Janmejay, Kamlesh & Himashu (2015)

Yang (2016) established network intrusion detection model based on RST. Anomaly Detection System based on Data mining was proposed in detecting suspicious behavior within port numbers, application layer data, and network data protocol. The results of the experiments indicate that the proposed model effectively detected attacks from attackers. Simultaneously, these methods can also protect network security. In Jun (2016) work, Rough Set was used as part of discretization (Semi Naives Scaler Algorithm) and rule analysis to segregate strong rule and weak rule based to be used in Bayesian classification as shown in Figure 2.7. The output from data mining part underwent discrimination of misinformation in Bayesian Network.

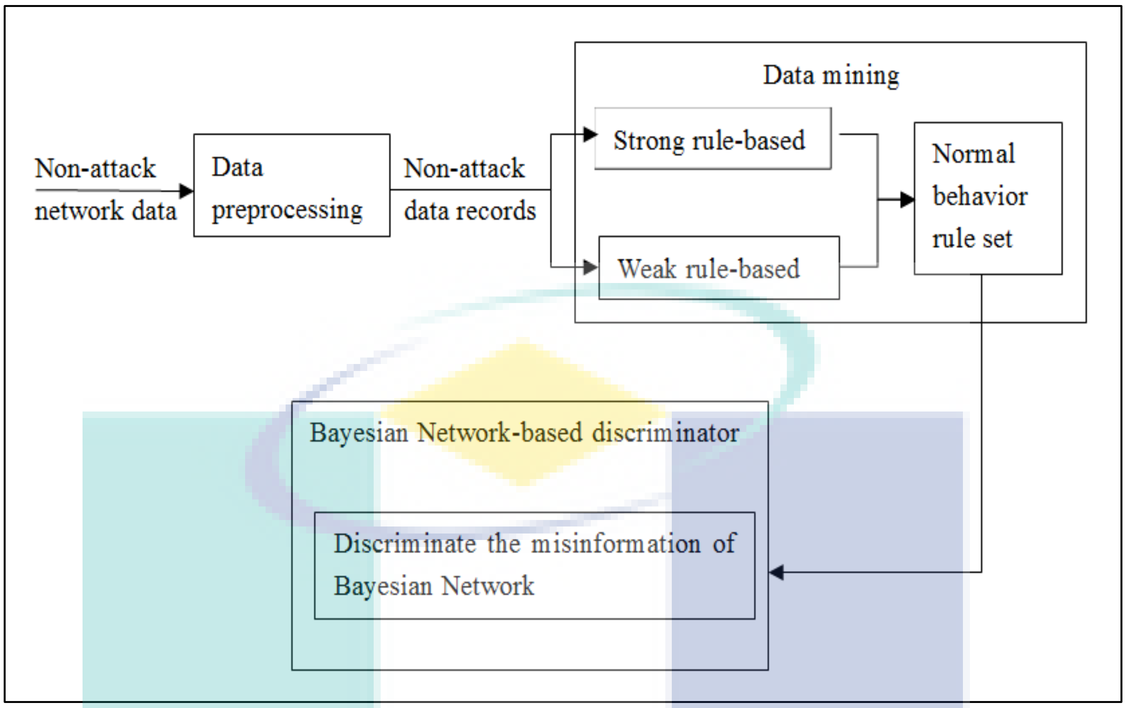

Figure 2.7 Training Stage of Anomaly Detection System Based on Data Mining
Source: Jun (2016)

In training experiment, 141 from 180 attacks were detected, with 78.3% detection rate. Weak rule based system was able to detect IP sweep and Port sweep satisfactorily. In testing, three different speeds were selected to evaluate the program: normal, tenfold and 1/10 speed. The results show that at tenfold speed, the system missed detection; while, at 1/10 speed, the system had more misinformation. However, the system was able to decrease attack misinformation after discriminator.

Panigrahi & Patra (2018) proposed a soft computing based technique to construct an intrusion detection model by five classifiers namely, Fuzzy Nearest Neighbour (FNN), Fuzzy-Rough Nearest Neighbour (FRNN), Fuzzy-Rough Ownership Nearest Neighbour (FRONN), Vaguely Quantified Nearest Neighbour (VQNN), and Ordered Weighted Average Nearest Neighbour (OWANN). The most relevant features in the input data were extracted through a preprocessing stage using wrapper subset evaluator. Finally, the performance of the model in terms of accuracy, detection rate and false alarm rate was evaluated on NSL-KDD intrusion dataset.

Figure 2.8 shows the model proposed by Panigrahi & Patra (2018). The preprocessing stage of dataset was conducted using Wrapper Subest Evaluator. Good subset using heuristic function was examined to find the highest state of evaluation. Then, feature selection was also performed through Wrapper Subest Evaluator. The selected features were trained and tested using Fuzzy Rough Classifiers to detect anomaly activities; and their performances were evaluated. From the results, it is shown that different classifier's strengths and weaknesses impacted each accuracy, precision and false alarm rate percentage; making it complex to have a single computation to improve IDS performance.

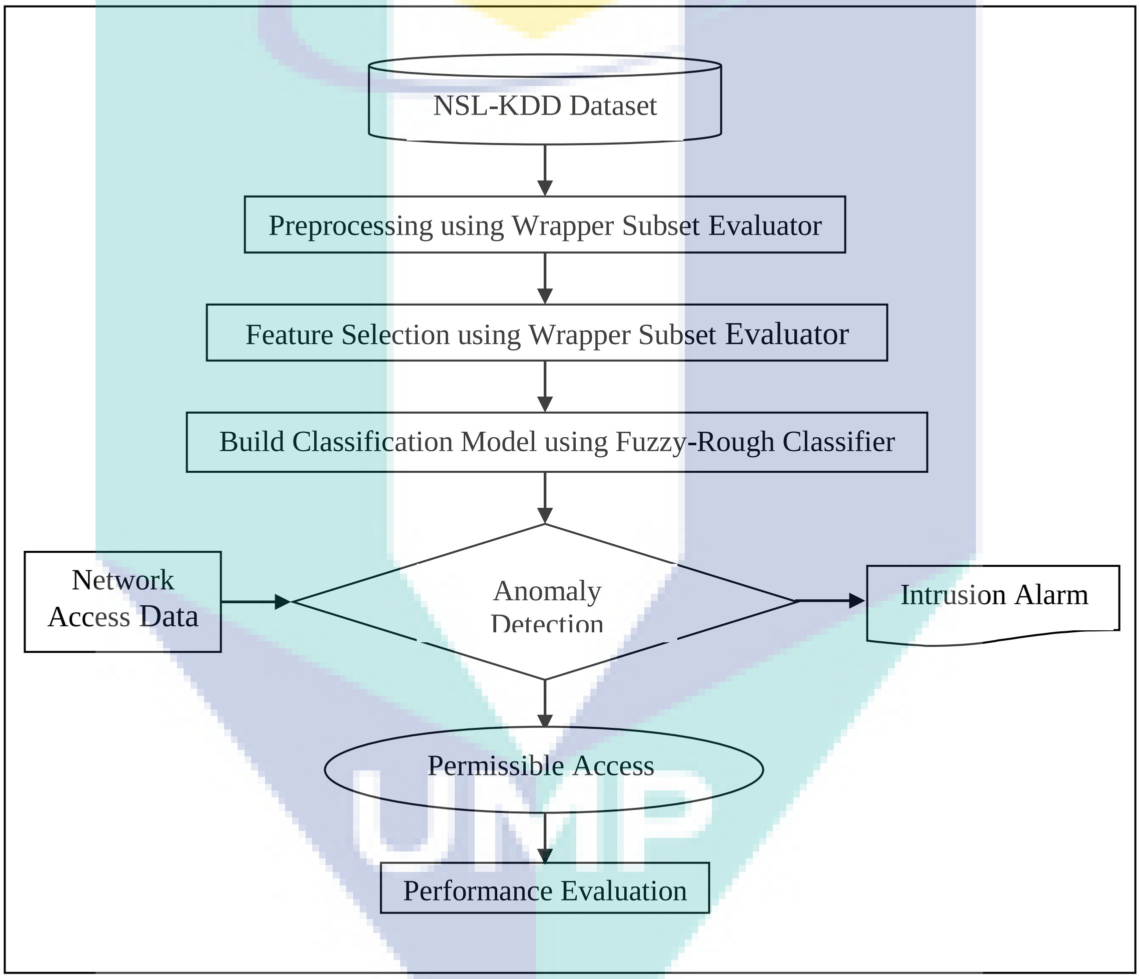
NSL-KDD Dataset
Preprocessing using Wrapper Subset Evaluator
Feature Selection using Wrapper Subset Evaluator
Build Classification Model using Fuzzy-Rough Classifier
Network Access Data
Anomaly Detection
Intrusion Alarm
Permissible Access
Performance Evaluation

Figure 2.8 Fuzzy Rough Set Based Network Intrusion Detection with Wrapper Subset Evaluator Model

Source: Ashalata & Manas (2018)

Table 2.3 depicts the summary of comparative previous research work analysis. From the discussion, it was discovered that there are various data mining classifiers such as RST (Yang, 2016), AIS (Shen et al., 2012), RST with Neural Network (Sadek et al., 2013), Fuzzy Nearest Neighbour (FNN), Fuzzy-Rough Nearest Neighbour (FRNN), Fuzzy-Rough Ownership Nearest Neighbour (FRONN), Vaguely Quantified Nearest Neighbour (VQNN), and Ordered Weighted Average Nearest Neighbour (OWANN) (Panigrahi & Patra, 2018). Meanwhile, several algorithms for feature selection were commonly used such as Q Learning algorithm (Sengupta et al., 2013), RST feature selection (Janmejay et al., Anazida et al., 2015) and Wrapper Subset Evaluator (Panigrahi & Patra, 2018). Different accuracy percentages were obtained from different attributes implementation depending on feature selection execution. Most classifiers show different capabilities and weaknesses in accordance to feature selection execution factor producing different impacts to classification performance in terms of accuracy, precision and false alarm rate percentages.

In comparative, research works of Hamid et al. (2015) gained the highest accuracy percentage at 99.95% with feature selection of 21 attributes. The feature selection was the factor of better classification percentage. Hamid et al. claimed that RST does not guarantee optimal reduct subset because of the overlap between classes of lower and upper approximation. Thus, Hamid et al. proposed new approach to enhance reduct generation process by adding union and voting of the attributes. Meanwhile, Yang (2016) discovered the lowest accuracy percentage in implementing RST into IDS classification is at 78.3% of 31 attributes. The issue that is still persistent today is how to select the number of intervals or bins and how to determine the bin width because these two problems are set manually by users. Precise numbers of interval and width affect classifier's accuracy.

Table 2.3 Other IDS Research Using Rough Set Technique

Author	Title	Year	Description	Accuracy %	Feature Selection	No of Attribute
Junyuang, Jidong & Hao	An Improved Artificial Immune System-Based Network Intrusion Detection by Using Rough Set	2012	•Proposed Artificial immune system (AIS) based network intrusion detection scheme . •RST used to optimize feature selection •Proposed scheme output is well performed compared to other schemes in accuracy detection.	98.25	Yes	41
Sadek, Soliman, & Elsayed	Effective Anomaly Intrusion Detection System based on Neural Network with Indicator Variable and Rough set Reduction.	2013	•New hybrid algorithm NNIV-RS (Neural Network with Indicator Variable using Rough Set for attribute reduct) is proposed. •Amount of computer resources used in attack detection is reduced. •RST is used in feature reduct.	96.7	Yes	40 (Training: 16799) (Testing: 9000)

Table 2.3 Continued

Author	Title	Year	Description	Accuracy %	Feature Selection	No of Attribute
Nandita Sengupta, JaydeepSen, JayaSil & MoumitaSaha	Designing of on line intrusion detection system using rough set theory and Q-learning algorithm	2013	• RST is used in employment of discernibility function, reduct and discretization generation. • Modify the Q-learning algorithm is done to learn different cut value in each conditional attribute. • Evaluate the reduct and accuracy to form the reward matrix. • Evaluate optimum cut values in each attributes using Modified Q matrix to achieve highest accuracy classification accuracy.	98.2	Yes	41
Hamid H. Jebur , Mohd Aizaini Maarof & Anazida Zainal	Enhancing Rough Set Theory Attributes Selection Of KDD Cup 1999	2015	• Enhance the reduct of class in encounter the problem of RST overlapping. • Achieved high and comparable RST rate of accuracy in the same dataset.	99.95	Yes	21

Table 2.3 Continued

Author	Title	Year	Description	Accuracy %	Feature Selection	No of Attribute
Janmejay, Kamlesh & Himanshu	Rough Set Approach for Feature Selection in IDS	2015	•RST is used in feature reduct and classification.	95.298	Yes	N/A
Yang Hui Jun	A Novel Rough Set Methodology and Machine Learning based Novel Network Intrusion Detection System: Theoretical Analysis and Applications	2016	•Output results shows that the proposed approach was efficient in attack detection.	78.3	Yes	31 (Training: 13,107) (Testing: 26,214)

Table 2.3 Continued

Author	Title	Year	Description	Accuracy %	Feature Selection	No of Attribute
Ashalata Panigrahi & Manas Ranjan Patra	Fuzzy Rough Set Based Network Intrusion Detection with Wrapper Subset Evaluator	2018	• Researcher using five classifiers; Fuzzy Nearest Neighbour (FNN), Fuzzy-Rough Nearest Neighbour (FRNN), Fuzzy-Rough Ownership Nearest Neighbour (FRONN), Vaguely Quantified Nearest Neighbour (VQNN), and Ordered Weighted Average Nearest Neighbour (OWANN).	97.513	Yes	41 (Total Record: 125973)

2.7 Overview of Intrusion Detection System

Starting in 1980, IDS is an active research area (Anderson, 1980). Six categories of intrusive activities and how these activities might be detected were recommended (Anderson, 1980). These recommendations led to the development of anomaly and misuse detection. IDS attempts to detect intrusion such as illegitimate uses, misuses and abuses of computer systems by using authorized user or external perpetrator (Biswanath & Heberlein, 1994).

Intrusion evaluation method is important in securing the networks. The method can be divided into four phases including preprocessing, analysis, response and refinement as depicted in Figure 2.9. Initially, IDS dataset undergoes preprocessing phase; next, the information from preprocessing phases is analyzed to determine the intrusion or normal event occurrence. Then, the response phase determines suitable action should be taken to match the event triggered. Finally, the refinement phase fine tunes the utilization and intrusion detected to have better IDS tool.

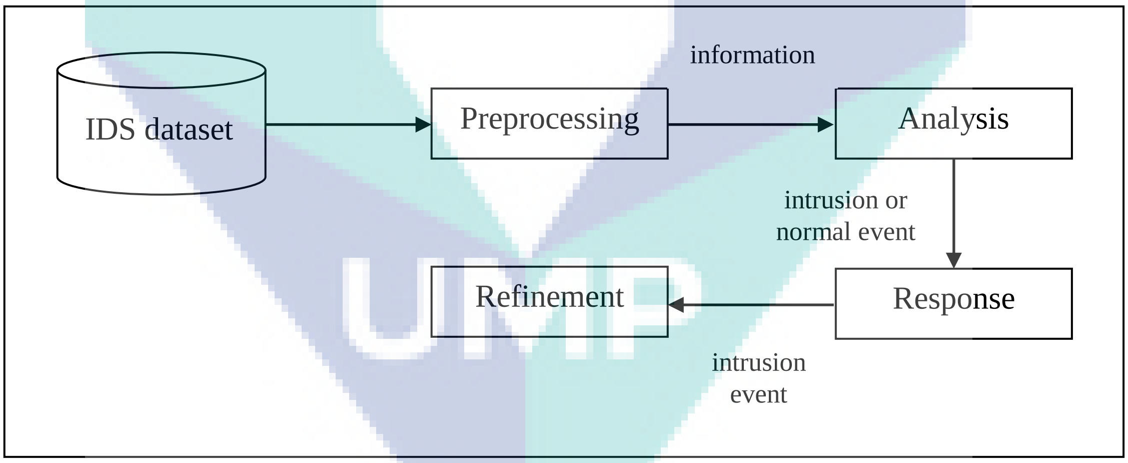

Figure 2.9 Intrusion Detection System Basic Phases

Source: Anderson, 1980

The details of each IDS phase are described as in the following:

a) Preprocessing

At this stage, the data is taken from IDS or IPS sensors. Statistics are sorted to form a pattern to be used in classification. Information is formatted and classified depending on schemas of analysis used.

b) Analysis

Information file is contrasted with the base of know-how. The facts file are to be logged as an event of intrusion; otherwise, the information file is dropped. The next record document is analyzed.

c) Response

The information is received inactively so caution is needed afterwards. The response can be performed to either automatically or manually after person-in- charge has manually analyzed the situation occurred.

d) Refinement

At this stage, fine tuning is finished in light of previous utilization and intrusion detected. This helps in decreasing false advantageous ranges and to have greater security tool like Cisco Threat Response (CTR). It assists the refining stage to ensure that the alert is valid via checking the inclination of the user to the attack. The rules are based on the detection such as signature detection, sample matching and misuse detection (Rizvi & Keole, 2015).

This thesis emphasizes on preprocessing and analysis IDS phases involving the construction of research objectives in improving the performance of accuracy, detection rate and reducing false alarm rate in IDS classification. IDS is classified into five categories which are Misuse, Anomaly, Host Based IDS, Network IDS and Hybrid IDS and described in Figure 2.10. The elaboration for each type is explained in the following sections.

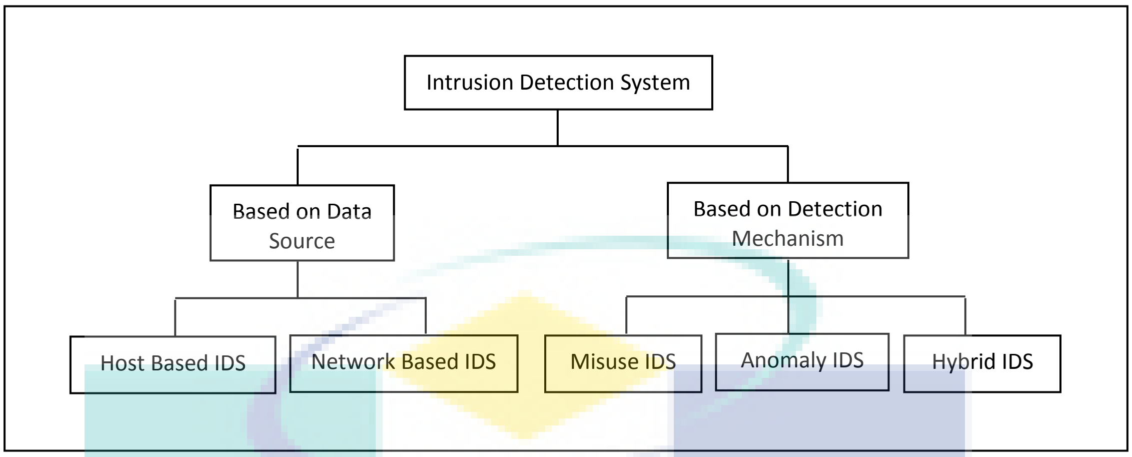

Figure 2.10 Types of Intrusion Detection System

a) Misuse Detection

Misuse detection usually detects intrusion signatures based on the rules set. A massive numbers of rule sets can be used to look for activities that maybe considered as intrusion state. The rules can be “*If-then*,” guidelines or certain based rules model. The activities might also be monitored live via monitoring device calls or later the usage of audit data (Kale et al., 2016).

b) Anomaly Detection

Anomaly detection defines variety of patterns of regular behaviors. Any abnormal elements observed from regular profiles are viewed as anomalies. Anomaly detection is tough to determine precisely from everyday profile. Thus, the detection generally suffers with higher false rate. Anomaly detection is divided to static and dynamic detections. In static detection, it is based on the assumption that there is an element of monitored device that does not change. If static portion of the device differs from its unique form, an error has occurred or an intruder has altered static portion of the system. Dynamic detection operates on audit archives or on monitored network traffic data. (Kale et al., 2016).

c) **Host Based IDS**

Host based IDS (HIDS) alludes to realize intrusions analyzed from information gathered from a single/individual host system. HIDS operator monitors the activities; for example, system integrity, utility activity, document changes, host based community traffic and machine logs. By utilizing frequent hashing tools, file timestamps, device logs, and video display units machine calls; local community interface offers the agent an understanding towards the present nation of nearby host. If unauthorized adjustments or activities detected, the user is alerted by a pop-up, additionally alerting the central management server, blocking off the activity, or a mixture of three mentioned. The option is primarily based on the policy installed in the local system. These host-based tactics are regarded as passive component. (Rizvi & Keole, 2015).

d) **Network Intrusion Detection Systems**

Network based totally IDS (NIDS) is utilized for monitoring and examining community site visitors to defend a machine from community's primarily based assaults where the statistics is going over the network. NIDS is capable to distinguish malicious actions and display the visitor's attacks network. NIDS comprises a variety of sensors to monitor packet movement. NIDS appears progressively at pastime packet, or close to actual time, to recognize intrusion patterns. The investigation of visitor's pattern in intrusion detection may be accomplished at the sensor, administration servers, or combination of both. NIDS tactic is regarded as e active component (Rizvi & Keole, 2015).

e) **Hybrid IDS**

Previous researches in IDS counseled that the intrusion detection competencies are elevated through a hybrid method involving signature (misuse) detection in anomaly detection. In hybrid approach, the signature detection method detects an acknowledged attack and the anomaly detection method detects a novel/unknown attack.

2.8 Technique of Attack Detection in IDS

Techniques of attack detection in IDS application are divided into seven categories as illustrated in Figure 2.11. The elaboration for each category is explained as follows:

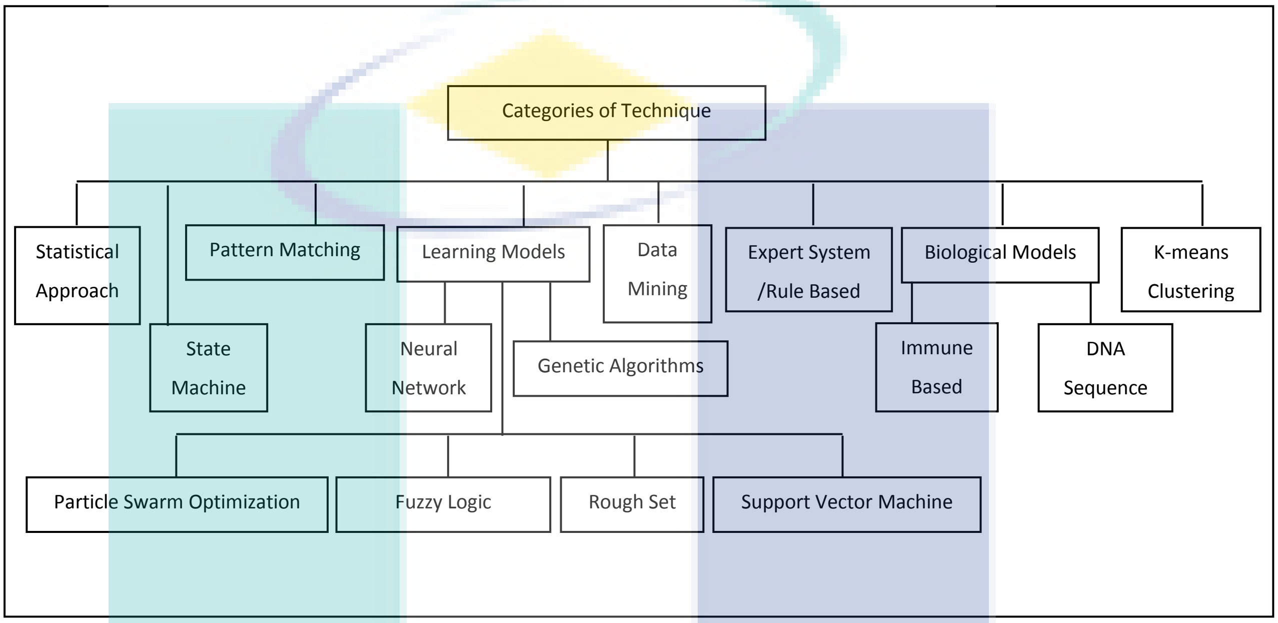

Figure 2.11 Categories of Techniques in IDS Application

2.8.1 Statistical Approach

This approach includes statistical comparison of specific event criteria setting. Attributes set are considered as variable number and known as "user login, logout, number of files accessed in a period of time, usage of disk space, memory", etc. (Elfeshawy & Faragallah, 2012). Data collected is tested for intrusion analysis through the use of statistical models.

2.8.2 Data Mining

Data mining techniques are widely used in the community and are host to build mostly in misuse detection model. IDS is expressed as a statistics evaluation process. Statistics mining method is used to automatically examine user's normal activity and intrusive behavior (Ghorbani et al., 2010). Data Mining is divided into four tasks

(Elfeshawy & Faragallah, 2012) such as Classification, Clustering, Regression, and Association Rule Learning.

2.8.3 Pattern Matching

Pattern matching based IDS procedure is regularly used in network to model, match and recognize the intrusion pattern based totally on the packet head, packet content or both (Ghorbani et al., 2010). Referring to new types and variety of attacks, the number of signatures is growing as it increases computational cost in pattern matching. Additionally, this approach does not detect new attacks.

2.8.4 Expert System / Rule Based

The expert system is based on previous set of rules describing the intrusions. The security related to events in audit trail is translated to *if-then-else* rules (Akbar et al., 2010). This approach develops statistical profiles of entities (as user, workstation and application program) and use statistical abnormal activities in detecting intrusions. Unfortunately, approach requires updates from System Administrator in remaining up-to-date. Lack of maintenances or updates is the weakness of expert system (Elfeshawy & Faragallah, 2012).

2.8.5 State Machines

This model consists of set of states, transitions and actions. Intruders launch the intrusion portrayed with the arrangement of objectives and transition accomplished to conquer the system (Akbar et al., 2010).

2.8.6 K-means Clustering

The algorithm uses input parameter *k*, then partitioning a set of *n* object into *k* cluster. The result of “intra-cluster” similarity is high; however, the “inter-cluster” similarity is low. The main purpose to employ this algorithm is to break up and crew statistics into normal and intrusion cases (Kumar et al., 2013).

2.8.7 Learning Models

Learning model incorporates getting to know competencies in intrusion detection process and the usage of synthetic learning technique. Previously, mastering methods are extensively employed in anomaly detection when considering that self learning strategies can afford to routinely structure the conducted subjects and opinions (Ghorbani et al., 2010). Several learning models are discussed in the following.

Previous IDS adapts the use of neural network, which consists of two important elements. First element is a system which monitors audit trails in recognizing intrusion signature. Second element is to examine consumer behavior, if abnormal state occurs then alarm is alerted (Ghorbani et al., 2010). Neural network is a class of machine learning algorithms used to classify data. A neural network consists of node and edge. The weight value defines how a node impacts adjoining nodes. A subset of nodes in the model is referred to as input nodes. Detection the usage of neural networks is three steps process.

Fuzzy Logic is a structure of many-valued good and approximate judgment, a substitute to constant and genuine reasoning. Fuzzy set idea was introduced by Zadeh in 1965. Fuzzy Logic is designed mathematically to characterize uncertainty and vagueness using tools in dealing with real world problems. Fuzzy Logic based devices should be able to detect types of intrusive undertaking PC networks as the rule base holds higher set of policies (Hassan, 2013).

Genetic Algorithm (GA) makes use of biological evolution as a critical questioning system. The proposed IDS based on GA includes two modules; for every work in an alternate stage, a set of classification is generated from network audit facts in offline condition. In the stage to detect intrusion, the generated policy is used to classify real time network connections. GA makes use of evolution and natural resolution using a chromosome as record structure and evolves chromosomes in the use of selection, recombination and mutation operator. Every chromosome role is encoded as bit, character or number. These positions could be referred to as gene (Hassan, 2013). Unfortunately, GA limitation notes that improper threshold value might easily lead to high false alarm rate in new intrusion detection (Ghorbani et al., 2010).

2.8.8 Biological Models

Previously, there is an anomaly detection models relating to biological principles named as Immune Based. IDS immune based is inspired by human immune system concept and is capable to perform similar tasks to innate and adaptive immunity. The normal behavior profile is generated by collecting services behavior gaining from audit data (Ghorbani et al., 2010).

Yu et al. (2001) proposed IDS based on DNA sequence. This research defines sequences of DNA for a computer system. The knowledge includes DNA characterization of human body and any anomaly in tissues that can reflect in DNA sequence. Any changes of behavior patterns in the computer system are traced to the change of DNA sequences that can be either a normal or intrusion event. Standard Backpropagation Neural Network is employed to train normal DNA sequence in network traffic. The system successfully detects UDP Flood attack based on DNA sequence in normal network traffic. Although convincing, this preliminary result needs to be expanded to define more complete DNA scheme in computing system (Ghorbani et al., 2010).

2.9 Related Works on Machine Learning of Intrusion Detection System

While investigating previous works conducted on Intrusion Detection System identified with machine learning methods, it comes to fore that there are three fundamental classifiers: Single classifiers, Hybrid classifiers and Ensemble classifiers.

2.9.1 Single Classifiers

Single classifier approach uses one-class classification in classifying data known as unary classification or class modeling. Among researches conducted on Single Classifier are Altwaijry & Algarny, (2012), Catania et al. (2012), Chi et al. (2012), Kausar et al. (2012), Li et al. (2012), Mohammed & Sulaiman, (2012) and Mukherjee & Sharma, (2012). Types of Single Classifier are discussed as in the following sections;

2.9.1.1 Fuzzy Logic (Fuzzy Set Theory)

Fuzzy Logic is utilized in reasoning. Its esteem ranges from 0 to 1; for example, raining is characteristic occasion (Zimmermann, 2001). Fuzzy Logic is viable and has much potential as a method. It manages human basic leadership and thinking. Fuzzy Logic utilizes “if then else” rules. It is utilized as a part of many building applications (Aburomman & Reaz, 2013), especially in anomaly detection. Fuzzy Logic is efficient in port scanning and probes; however, it consumes high resources (Kaur et al., 2013).

2.9.1.2 Genetic Algorithms

Genetic Algorithm selection capability refers to criteria of performance (Kamran, 2009). GA is inspired by biologically heuristic search. IDS collecting traffic information employs GA; then, gathers information at normal state or intrusion state at present (Borgohain, 2012).

2.9.1.3 K-Nearest Neighbor

K-Nearest Neighbor (k-NN) is old and simple approach in sample classification (Manocha & Irolami, 2007). Parameter *k* is essential to create k-NN classifier. The *k* value changes impact the different performances. k-NN calculates a rough distance between two different points; for instance, base learning and it difference from inductive approach.

2.9.1.4 Support Vector Machine (SVM)

Support Vector Machine is superior by setting up a hyper plan. This approach classifies the statistic into groups; then, divides the records into two portions to solve problem of vector and quadratic programming (Horng et al., 2011).

2.9.1.5 Artificial Neural Networks

Artificial Neural Network (ANN) processes and classifies data inspired by the neurons of human brain. Multilayer Perceptron is basically used in neural community architecture. ANN is fast learning and capable to analyze "non-linear information set" with "multi variables" (Wu & Huang, 2010).

2.9.1.6 Decision Trees (DT)

This approach classifies a pattern generated from more than few decisions. The first selection helps the second selection; until it finally becomes a tree structure. The classification of sample starts with root node and ends with quit node which is additionally called leaf node. Each quit node (leaf node) presents a class of classification (Peddabachigari et al., 2007).

2.9.2 Hybrid Classifiers

Hybrid architecture is designed with several heterogeneous methods, which complement each other in improving the performances. Hybrid classifiers are capable to enhance anomaly and misuse detection (Govindarajan & Chandrasekaran, 2011) to combine Host Based IDS (HIDS) and Network Based IDS (NIDS).

Bahrainian & Dangel (2013) introduced a hybrid method of sentiment lexicons usage with a classifier of machine learning to detect opinions polarity in consumer and product area. Gautam & Yadav (2014) proposed machine learning technique analysis of semantic to classify sentences and reviews of product based on twitter data using WordNet to improve accuracy. Gokulakrishnan et al. (2012) examined different classifier performances such as Naïve Bayesian, Sequential Minimal Optimization (SMO), SVM and Random Forest in classifying Twitter data. Meanwhile, Chen et al. (2014) proposed a technique to classify student's Twitter data into several categories. However, the process of generation category is static. Furthermore, the static process results in the limitation of classifiers to improve accuracy performance.

2.9.3 Ensemble Classifiers

Ensemble classifier is a group of individual classifiers which cooperate with each other to train the dataset in supervised classification. Ensembler classifier combines several single classifiers to generate an improved output (Majidi et al., 2008). Refer to Table 2.4 for other researches carried out in 2006, 2011 and 2012.

Kumar et al. (2013) proposed a hybrid intrusion detection system. They used several classification techniques in their proposed method. They combined Bayes Theorem, Naive Bayes Classifier and K-Means Clustering to detect intrusions.

2.10 Evaluation of IDS Classification

In part of IDS result, Julisch & Dacier (2002) discussed about the possibility of IDS to have numerous false alarms. IDS is not completely accurate; thus, it possible to have false negative alarms that may not detect several attacks. In IDS, to ensure efficiency the following five parameters are proposed to be achieved. First is (Debar, 2009) **Accuracy,** which manages proper attack detection and false alarm absence. If inaccuracy occurs, IDS flag unauthorizes action either intrusive or anomalous. Second is **Performance** which is rated at processing audit event. If IDS is lack of performance, then it is not possible to detect intrusion in real time. Third is **Completeness** which refers to IDS property in attack detection. Incompleteness happened when IDS is not capable to detect intrusions. Fourth is **Fault Tolerance** which determines IDS resistant to attacks, especially Denial of Service attacks. Most IDS runs on operating systems or hardware that is exposed to vulnerable and attacks. The final parameter is **Timeliness** which determines how IDS should perform the analysis to immediately alert security officer before more damages occur to prevent intruders from subverting the audit sources.

2.11 KDD Cup 99 Description

KDD Cup 99 dataset was used in KDD Cup 99 Classifier-Learning Competition. Table 2.4 shows number of attributes of KDD Cup 99 dataset that consist of 41 network attributes as condition attribute and one attack type attribute as decision attribute.These sets of training and test data were made available by Stolfo and Lee (http://kdd.ics.uci.edu/databases/KDD Cup99/task.html, 1999) and consisted of a preprocessed version of 1998 DARPA Evaluation Data.

Table 2.4 KDD Cup 99 Attributes

No	Attribute	No	Attribute	No	Attribute	No	Attribute
1	duration	12	logged_in	23	count	34	dst_host_same _srv_rate
2	protocol_ type	13	num_ compromised	24	srv_count	35	dst_host_diff_ srv_rate
3	service	14	root_shell	25	serror_rate	36	dst_host_same _src_ port_rate
4	flag	15	su_attempted	26	srv_serror_rate	37	dst_host_srv_ diff_ host_rate
5	src_bytes	16	num_root	27	rerror_rate	38	dst_host_ serror_rate
6	dst_bytes	17	num_file_ creations	28	srv_rerror_rate	39	dst_host_srv_ serror_ rate
7	land	18	num_shells	29	same_srv_rate	40	dst_host_ rerror_rate
8	wrong_ fragment	19	num_access_files	30	diff_srv_rate	41	dst_host_srv_ rerror_ rate
9	urgent	20	num_outbound_ cmds	31	srv_diff_host_ rate		
10	hot	21	is_host_login	32	dst_host_count		
11	num_failed _logins	22	is_guest_login	33	dst_host_srv_ count		

Source: http://kdd.ics.uci.edu/databases/KDD Cup99/task.html (1999)

The KDD Cup 99 Classifier-Learning Competition team had performed particularly well in Intrusion Detection Evaluation Program of that year. They used data mining at preprocessing stage to extract characteristic intrusion features from raw TCP/IPn audit data. The original raw data training of 4 gigabytes of compressed binary tcpdump data was obtained from the first 7 weeks of network traffic at Massachusetts Institute of Technology (MIT). Data were preprocessed with feature construction framework Mining Audit data for automated model for Intrusion Detection (MADAM ID) to produce about 4,898,431 and 311,029 records in set of training and testing records.

Furthermore, it become the de facto benchmark in evaluating merit performance of IDS (Kabir et al., 2018). Attacks in the dataset are segregated to Denial of Service (DOS), Remote to Local (R2L), User to Root (U2R) and Probing categories. Dataset is attached in Appendix A.

2.12 Conclusion

In conclusion, this thesis addresses the problem of discretization limitation and generation rules as discussed previously. By enhancing classification techniques specifically in discretization algorithm and strategy of rule generation to improve performance, attack detection can be improved and evaluated against KDD Cup 99 dataset. This research does not focus on IDS model; instead it focuses on the contribution of enhancement technique that is constructed (involving discretization algorithm and strategy of rule generation) to improve classification performance.

CHAPTER 3

RESEARCH DESIGN AND METHODOLOGY

3.1 Overview

This chapter presents research methodology, research framework and proposed technique for classification in IDS application. Further elaboration is made on the enhancement of Frequency Binning discretization and rule generation strategy in determining significant rules to improve IDS application classification performance. These performances are in terms of accuracy, detection rate and false positive alarm rate decrement.

3.2 Research Methodology Framework

Research methodology framework for the enhancement classification technique of IDS application in Rough Set involves four phases. Those include are preliminary study, designing the proposed technique, evaluating the result of proposed technique and writing the thesis which are explained in next sections as illustrated in Figure 3.1.

3.2.1 Preliminary Phase

Preliminary phase began with preliminary study comprised of overview on data mining and classification techniques, RST approach, overview of IDS phases, related work and limitation of discretization algorithm and generation rule, related works on IDS model and overview of IDS benchmark dataset. Afterwards, research problem and research gaps of the study were obtained, followed by the formulation of aim and objectives of the study. Data collection of KDD Cup 99 benchmark data was gathered and analyzed through preprocessing stage. The preprocessing stage involved eliminating redundant, irrelevant and misleading features to encourage IDS performance percentage. At the beginning, the dataset was loaded into the classifier to determine suitable dataset portion that can be processed by the machine based on its capability to process the dataset. This stage continued with Literature Review to have in-depth understanding about the problem and to find the based of the proposed solution.

3.2.2 Design Phase

This stage is a process of designing the proposed classification technique enhancement for IDS application. The steps started off by designing the enhancement of Frequency Binning discretization and rule generation strategy to determine rules that are significant to improve IDS classification performance in terms of accuracy, detection and false positive alarm. Dataset underwent standard Rough Set Theory classifier including preprocess, discretization, reduct, generation rules and classification processes. The results of standard RST accuracy classification were compared to the results of proposed technique. The proposed works from this stage were evaluated in the evaluation phase to validate and compare the proposed works.

3.2.3 Evaluation Phase

The preprocessed IDS benchmark dataset, which is KDD Cup 99 dataset, was trained and tested against the proposed work then the results were evaluated. If the result analysis was found satisfactory (increment in accuracy classification, better attack detection rate percentage and low false alarm rate) compared to other related works; then, the next stage was to proceed with thesis writing and end the research stage. However, if the result was found to be unsatisfactory, then the proposed technique was redesigned and redeveloped. IDS parameter and ROC tool were involved in this validation process. ROC measurement is the common tool used to measure IDS accuracy classification percentage. ROC graph was plotted to confirm the validation based on the gathered results. This phase is elaborated in details in Chapter 4. Chapter 4 explains the experiments phase and discusses the results by making comparisons with other techniques, the evaluation and validation steps taken within this research.

3.2.4 Writing Phase

This final stage requires writing and completing all chapters in the thesis including Introduction (research background, research problem and gap, research objective and scope), Literature Review (reviews pertaining to research study and related works), Research Design and Methodology (research framework, proposed technique and algorithm) Result and Discussion (implementation and evaluation results discussion) and Conclusion (summary of research works and revisit research objective and contribution of the study). This thesis focuses on proposing a discretization technique based on Frequency Binning and generating significance rules strategy that is suitable with IDS application accuracy. The accuracy, detection rate and false alarm rate percentage of the proposed technique were compared to RST standard and other techniques. Finally, the findings were evaluated and validated.

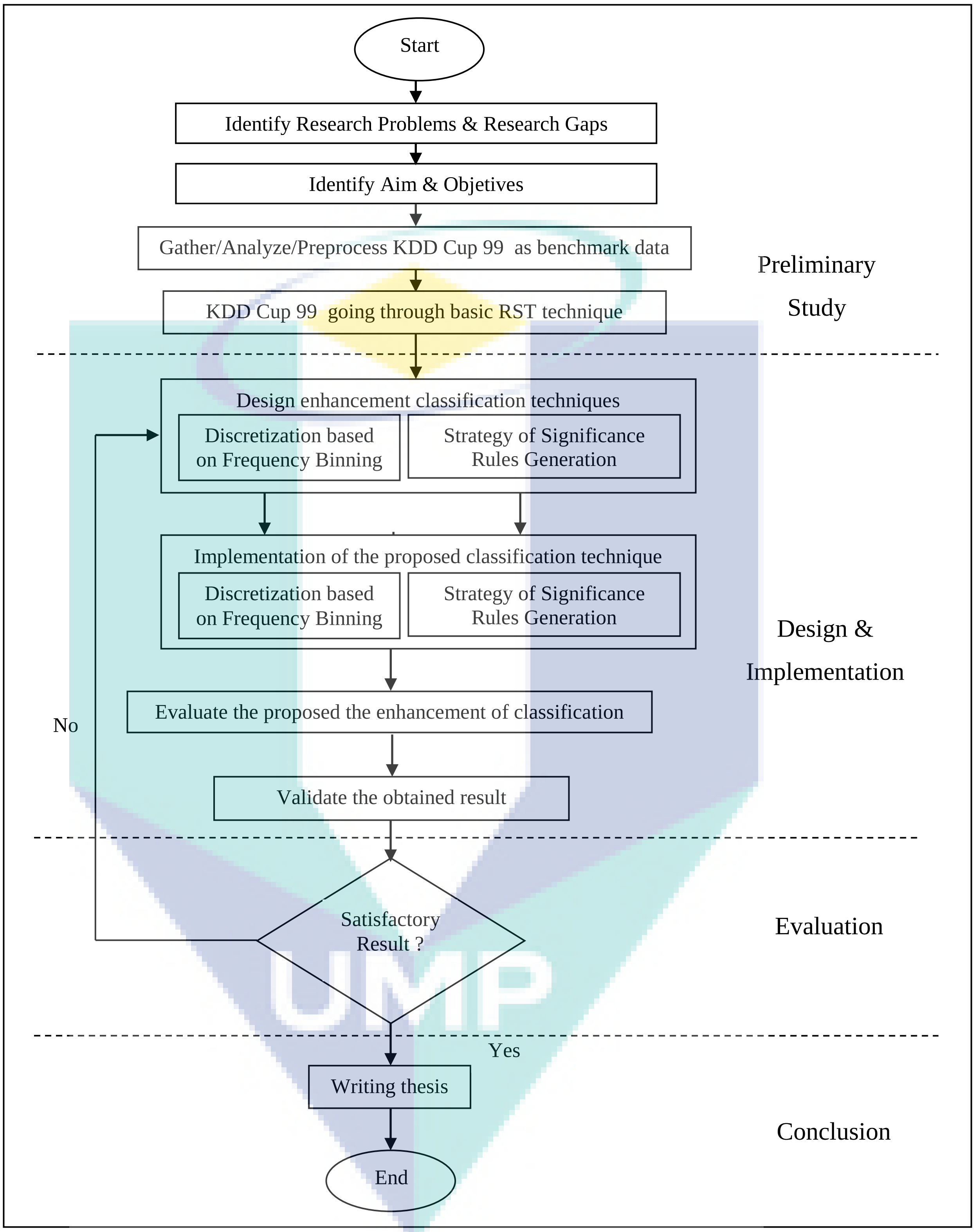
Start
Identify Research Problems & Research Gaps
Identify Aim & Objetives
Gather/Analyze/Preprocess KDD Cup 99 as benchmark data
KDD Cup 99 going through basic RST technique
Preliminary Study
Design enhancement classification techniques
Discretization based on Frequency Binning
Strategy of Significance Rules Generation
Implementation of the proposed classification technique
Discretization based on Frequency Binning
Strategy of Significance Rules Generation
Design & Implementation
Evaluate the proposed the enhancement of classification
No
Validate the obtained result
Satisfactory Result ?
Evaluation
Yes
Writing thesis
Conclusion
End

Figure 3.1 Research Methodology Framework

In general, RST framework is illustrated in Figure 3.2. This figure represents the design and implementation phase of research methodology framework. Figure 3.2

illustrates RST standard steps including the proposed enhancement on discretization and new strategy in generating rules with the purpose to increase accurate performance of IDS application in detecting attack from outsiders. As shown in the figure, initially, the dataset went through preprocessing step; whereby record of missing values were discarded by eliminating noise data. Then, decision table was created from raw dataset. The dataset experienced the standard of Equal Frequency Binning Discretization algorithm (Figure 3.2). Afterwards, the data split into *m* training and *n* testing categories. In this study, the dataset was split into 70% of training and 30% of testing records. Reduct was generated by using Genetic Algorithm as a standard reducer algorithm that is available in RST. Rules are created for classification process. As from here, all of these steps utilize standard Rough Set Theory. In the first phase of thorough analysis and evaluation, all reducts were based on Minimal Cardinality and Core Attributes algorithm. In this study, only 31 out of 42 attributes were considered significant based on two parameters of Minimal Cardinality and Core Attributes which to be processed in the proposed enhancement technique.

In this study, two steps (discretization and rule generation) were enhanced (Objective 1) and integrated into RST standard (Objective 2), refer to Figure 3.2. Simplification decision table was developed based on attributes of reducts with minimal cardinality and core attributes. Simplification decision table had the same process as previous stage, which was to create decision table and map the data. Then, dataset experienced the Proposed Frequency Binning Discretization. Dataset split into 70% of training and 30% of testing records. Reduct was generated using RST standard reducer of Genetic Algorithm. Afterwards, rules generated were analyzed to increase the accuracy performance of classification.

Finally, classification process was executed. The results of classification performance between decision table and simplification decision table were compared. Experimental results were analyzed and discussed. The results were validated (Objective 3) in a measurement of accuracy and fault tolerance variable. The significant reduced algorithm then proposed the enhancement of binning discretization and strategies to generate significant rule. They are further elaborated in the next sub sections.

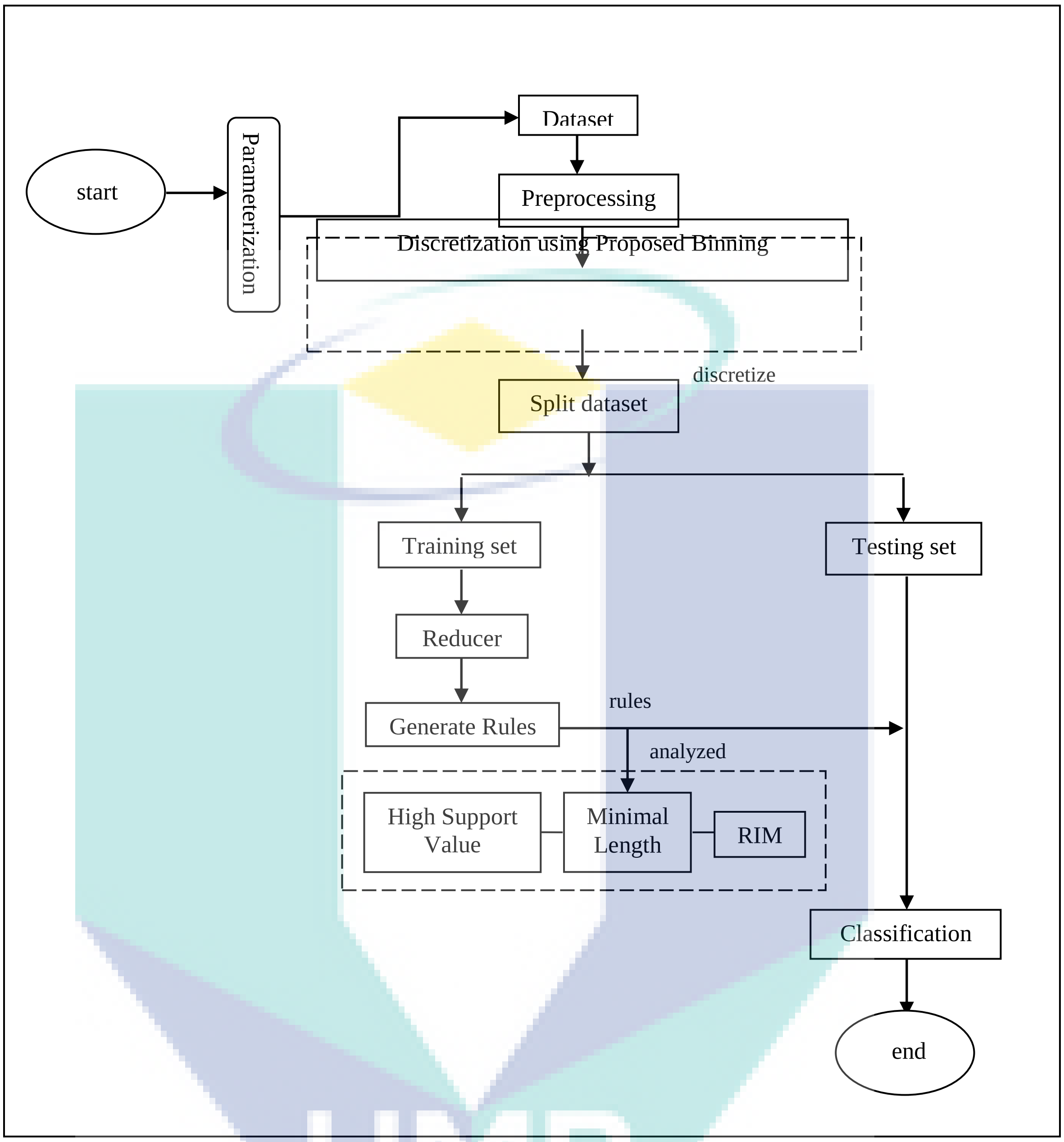

Figure 3.2 Steps of Rough Set Theory with Proposed Enhancement of Discretization Technique and Rule Generation.

3.3 Proposed Binning Discretization

The enhancement of Binning Discretization algorithm was proposed and the significance rule generation strategy was enhanced. The dataset underwent preprocessing stage of removing missing or irrelevant values so that it can proceed with IDS of RST classification. The purpose of this stage is to increase data quality; thus, the need for data cleaning increases significantly (Rahm, 2000). Then, the data was experimented with Proposed Binning Discretization to attain discrete data. The dataset was split into 70% training, whereby it was used in reduct and rules generation stage; and 30% testing records was used in classification stage. RST reducer generates the reduct by using Genetic Algorithm. In the reduct stage, analysis was performed to cope with core attribute and to identify the reduct with minimal cardinality as significant reduct to be forwarded to the next simplification decision table classification. The classification was processed using the rule obtained in previous stage. In enhancing Rough Set Theory attribute reduct model, two approaches were applied. Firstly, reduct with minimal cardinality as suggested by (Suresh & Anima , 2012)

$$\text{Re}d_{\min} = \{\text{R} \in \text{Re}d \mid \vee \text{A}^{'} \in \text{Red}, |R| \leq | R^{'} |\} \qquad 3.1$$

Secondly, intersection of reducts is the core reduct. Core reduct is the attribute element that cannot be eliminated as indispensable attributes. Core reduct can be defined as in the following (Suresh & Anima , 2012)

$$\text{Core}(C) = \cap \text{Re}d \qquad 3.2$$

Universally, discretization is a process of searching for partition of attribute domains into intervals and unifying the values over each interval. Discrete values are intervals of number which are more concise to represent and specify. Additionally, they are easier to utilize and comprehend as they are closer to knowledge level representation than continuous values. Discretization involves searching for cuts that determine intervals. All values that lie within each interval are mapped to the same value. As a result, it converts numerical attributes which are considered symbolic. The search for cuts is performed on the internal integer representation of input decision system. A discretization also (Pawlak & Skowron, 2007) replaces value sets of conditional real-valued attributes with intervals. The replacement ensures that a consistent decision system is obtained. It assumes a given consistent decision system by substituting the original values of objects in decision table by unique names of the intervals comprising these values. This substantially reduces the size of the value sets of real-valued attributes. Discrete values have important roles in data mining and knowledge discovery. Rules with discretize values are normally shorter and more understandable. Discretization can also improve the accuracy of performance when using discrete values rather than continuous values (Hacibeyoglu et al., 2011).

Equal Frequency Binning is one of the Rough Set Theory discretization algorithms (Dougherty et al., 1995). Binning algorithm is a method to discretize continuous valued attribute by creating specified number of bins. The bins ae created using equal frequency approach where an equal number of continuous value is placed within each bin. Then, each bin is associated with a distinct discrete value.

Decision table is comprised of 4 tuple as follows; (Dougherty et al., 1995)

$S = <U, Q \cup \{d\}, V, f>$, where

U : a finite set of N objects $\{x_1, x_2, \ldots\ldots\ldots, x_n\}$;

Q : a finite set of n condition attributes$\{q_1, q_2, \ldots\ldots\ldots, q_n\}$; (a nonempty set), and d is decision attribute ;

$\mathrm{V} = \bigcup_{q \in Q} \mathrm{V}_q$, where V_q is a domain of the attribute q.

$f : \mathrm{U} \times \mathrm{Q} \cup \mathrm{d} \rightarrow \mathrm{V}$ is total decision function called information function like $f(\mathrm{x}, \mathrm{q}) \in \mathrm{V}_\mathrm{q}$ for every $q \in \mathrm{Q} \cup \mathrm{d}, \mathrm{x} \in \mathrm{U}$.

The decision corresponds with able of finite data. Columns are labeled by attributes, rows by objects and entry in column q_j and row x_i has the value $f(x_i, q_j)$. Each row in the table describes the information about some objects in S.

Assume $\mathrm{V}_q = [l_q, r_q) \subset R$, where R the set of real numbers is, and assume that S is consistent decision table. The following notion and description about discretization is referred to reference (Dai & Li, 2002).

Definition 1 Any pair (q, c), where $q \in \mathrm{Q}$ and $c \in \mathrm{R}$,defines a partition of V_q into left-hand-side and right hand-side interval. The pair (q, c) is called a cut on V_q.

Attribute $q \in \mathrm{Q}$, $D_q = \{(q, c_1^q), (q, c_2^q), \ldots, (q, c_{k_q}^q)\}$ is comprised of all cuts, where

$k_q \in N$, and $l_q = c_0^q < c_1^q < c_2^q < ... < c_{k_q}^q < c_{k_q+1}^q) = r_q$, defines a partition on V_q into subintervals i.e. $V_q = [c_0^q, c_1^q) \bigcup [c_1^q, c_2^q) \bigcup ... \bigcup [c_{k_q}^q, c_{k_q+1}^q)$. Hence, any set of cuts on condition attributes $D = \bigcup D_a$ transforms decision table S into discrete decision table $S^D = < U, Q \bigcup \{d\}, V^D, f^D >$, where $f^D(x,q) = i \Leftrightarrow f(x,q) \in [c_i^q, c_{i+1}^q)$, and $x \in U, i \in \{0,1,...,k_q\}, q \in Q$.

After discretization, the decision table is replaced with new one. Furthermore, different sets of cuts develop different simplification decision table. Discretization work process is elaborated in equivalence class section. Obviously, the discretization process is related with information loss. Normally, the purpose of discretization undertakings is to decide minimal set of cuts required from a given decision table and to maintain discernibility.

Equal Frequency Binning is a benchmarking discretization algorithm for the proposed enhancement discretization algorithm. The proposed Binning Discretization, attribute $q \in Q$, $D_q = \{(q, c_1^{qa}), (q, c_2^{qb}), ..., (q, c_{k_q}^{qn})\}$ is composed of cuts, where $k_q \in N$, and $l_q = c_0^{qa} < c_1^{qb} < c_2^{qc} < ... < c_{k_q}^{qn} < c_{k_q+1}^{qn+1}) = r_q$, defines a partition on V_q into subintervals which is $V_q = [c_0^{qa}, c_1^{qa}) \bigcup [c_1^{qb}, c_2^{qb}) \bigcup ... \bigcup [c_{k_q}^{qn}, c_{k_q+1}^{qn+1})$. Thus, set of cut in condition attributes $D = \bigcup D_a$ transforms decision table *S* into discrete decision table $S^D = < U, Q \bigcup \{d\}, V^D, f^D >$, where $f^D(x,q) = i \Leftrightarrow f(x,q) \in [c_i^{qa}, c_{i+1}^{qn})$, and $x \in U, i \in \{0,1,...,k_q\}, q \in Q$. The proposed algorithm can be referred to Figure 3.3. Initially, q must be equal to empty value. Then, *q* values is sorted $\in Q$, *Q* is a finite set of *n* condition q attributes. Lines 3.0 to 3.2 are the process of proposed discretization algorithm. Value of V_q (domain of attribute *q*) is computed to generate cumulated different class frequencies for each subinterval, between cumulated class frequencies *D*. Afterwards, class frequencies are transformed to decision table *S* to S^D.

```
Input: Dataset from KDD Cup 99
Output: Disretize values
1   Start
2       Set $q = 0$
3       Sort the attribute values: $q \in Q$

    // Start discretization process
4       For each subinterval $V_q$
5           Compute $V_q = [c_0^{qa}, c_1^{qa}) \cup [c_1^{qb}, c_2^{qb}) \cup \ldots \cup [c_{k_q}^{qn}, c_{k_q+1}^{qn+1})$
6           Assign cumulated class frequencies $D = \cup D_a$
7           Assign decision table $S = S^D$
8       End for
9   End
```

Figure 3.3 Proposed Binning Discretization Algorithm

3.4 Enhanced Strategy to Generate Significance Rules

Four condition attributes from KDD Cup 99 dataset, which are *protocol_type, service, flag* and, *src_bytes,* and one decision attribute which is *attack* are shown in Table 3.1. They were taken as samples to elaborate step by step process of rule generation. On the other hand, Figure 3.4 shows the steps of generating rules process including equivalence class, discernibility matrix, reduct and rule generation. Equivalence class searches for hidden object relation to determine significant related for classification process. Discernibility matrix searches for different objects; meanwhile, reduct generates unique set of objects. The results show that the objects are significant to be employed in generating rules to increase accuracy classification performance.

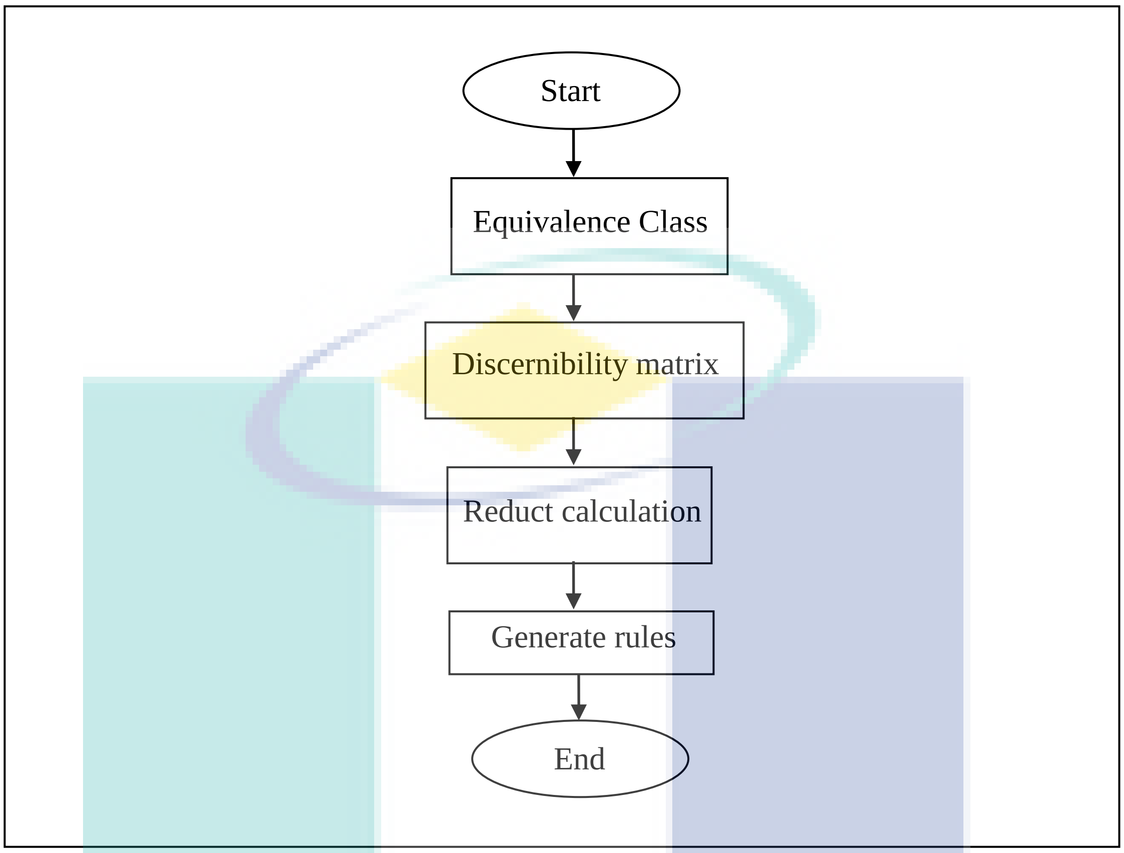

Figure 3.4 Steps of Rules Generation

3.4.1 Equivalence Class

The aim of equivalence class is to search for object's indiscernibility relation to courage better impact for accuracy classification performance. Next subsection details out equivalence class process, discernibility matrix, reduct and the rule generated by employing KDD Cup 99 dataset sample based on Table 3.1.

Table 3.1 Sample of Decision Table Taken From KDD Cup 99 Dataset.

Case	**Attributes**				**Decision**
	Protocol_type	**Service**	**Flag**	**Src_bytes**	**Attack**
1	tcp	http_443	SF	239	normal
2	udp	finger	S0	290	neptune
3	tcp	ftp_data	SF	1377	ipsweep
4	tcp	ftp_data	SF	424	ipsweep

Case object of dataset is mapped to equivalence class as shown in Table 3.2. Case 1 is mapped to E1, case 2 is mapped to E2, case 3 and case 4 are mapped to E3. Both case 3 and 4 have the same object values in condition and decision attributes, except in value of *src_bytes*, which are case 3 with 1377 and case 4 with 424 respectively. Consequently, case 3 and 4 are mapped into the same E3 class but with different cases. They are mapped into different cases (case 3 to (E3, 1) and case 4 to (E3, 2)) because of they have the same attribute values except for *src_bytes*.

Table 3.2 Sample of Equivalence Class Based on Decision Table

Class	**Protocol_type**	**Service**	**Flag**	**Src_bytes**	**Attack**
E1	tcp	http_443	SF	239	normal
E2	udp	finger	S0	290	neptune
E3,1	tcp	ftp_data	SF	1377	ipsweep
E3,2	tcp	ftp_data	SF	424	ipsweep

Then, discretization stage is generated from equivalence class as in Table 3.3 by converting to numerical representation. Each value of attributes is allocated in the class as protocol_type, service, flag, src_bytes and attack with their own object value representation. For example, Table 3.3 indicates class *"attack"* where three different types of attack are represented as 1: *normal*, 2: *neptune* and 3: *ipsweep*.

Table 3.3 Relation Mapping to Discretize

Class	**Representation**
Protocol_type	1: tcp, 2: udp
Service	1: http_443, 2: finger, 3: ftp_data
Flag	1: SF, 2: s0
Src_bytes	1: 239, 2: 290, 3: 1377, 4:424
Attack	1: normal, 2: Neptune, 3: ipsweep

As shown in Table 3.4, the equivalence class from Table 3.2 is mapped again based on discretized data. Value of *a,b,c* and *d* represents condition attributes of *protocol_type*, *service*, *flag* and *src_bytes*. Meanwhile, value of *dec* represents decision attribute of attack. Object values of *1, 2* and *3* represent discretization process as shown in Table 3.3. Discretization process converts continuous value to discrete value so that concrete object can be processed in the classification process.

Table 3.4 Equivalence Class after Discretization

Class	a	b	c	d	dec
E1	1	1	1	1	1
E2	2	2	2	2	2
E3,1	1	3	1	3	3
E3,2	1	3	1	4	3

3.4.2 Discernibiliy Matrix

Table 3.5 maps discernibility matrix to determine different objects carried out to generate the reduct. Condition classes of E1, E2, (E3,1) and (E3,4) are mapped into E1, E2, E3 and E4. The decision class of dec is mapped as *f*. Values of *a,b,c* and *d* are mapped into discernibility matrix. The discernibility matrix checks individual discernibility across classes as indicated in Table 3.4. For example in (E1,E1) are of the same value so that the value is *x*. The value across (E1,E2) are discerned in each *a,b,c,d* condition attribute; thus, the value in (E1,E2) are *{a,b,c,d}*. Meanwhile, in (E1,E3), the discerns are in condition b and d so that the value in (E1,E3) are *{b, d}*. All the discernibility values are carried out in reduct generation.

Table 3.5 Sample of Discernibility Matrix Based on Equivalence Class

	E1	E2	E3	E4	*f*
E1	x	{a,b,c,d}	{b,d}	{b,d}	{a,b,c,d}
E2	{a,b,c,d}	x	{a,b,c,d}	{a,b,c,d}	{a,b,c,d}
E3	{b,d}	{a,b,c,d}	x	{d}	{a,b,c,d}
E4	{b,d}	{a,b,c,d}	{b,d}	x	{a,b,c,d}

3.4.3 Reduct

Reduct generation results from discerned object in equivalent class. The outputs of reduct are gathered from discernibility matrix values. Therefore, based on Table 3.5, reduct generation are:

{a,b,c,d}
{b,d}
{d}

3.4.4 Generate Rules

Finally, rule generation classifies testing dataset in IDS performance in terms of accuracy, detection and false positive alarm. The rule derivations are based on condition attributes possibilities in reduct of *{a,b,c,d}*, *{b,d}* and *{d}*. They are mapped to decision attribute *{e}*, with condition attribute (value) AND condition attribute (value) => decision attribute (value). Knowledge representation with decision is representation of *a (protocol type), b (service), c (flag), d (src_bytes) and e (attack)*. Meanwhile, data from decision table uses actual name of attributes and object values.

Table 3.6 Rules Derivation Based on Reduct Generation

Knowledge Representation with Decision	Data from Decision Table
a(1) AND b(1) => e(1)	protocol_type (1) AND service(1) => Attack(1)
a(2) AND b(2) => e(1)	protocol_type (2) AND service(2) => Attack(2)
a(1) AND b(3) => e(1)	protocol_type (1) AND service(3) => Attack(3)
a(1) AND c(1) => e(1)	protocol_type (1) AND flag(1) => Attack(1)
a(2) AND c(2) => e(2)	protocol_type (2) AND flag(2) => Attack(2)
a(1) AND c(1) => e(3)	protocol_type (1) AND flag(1) => Attack(3)
a(1) AND d(1) => e(1)	protocol_type (1) AND src_bytes(1) => Attack(1)
a(1) AND d(2) => e(2)	protocol_type (2) AND src_bytes(2) => Attack(2)
a(1) AND d(3) => e(3)	protocol_type (1) AND src_bytes3) => Attack(3)
a(1) AND d(4) => e(3)	protocol_type (1) AND src_bytes(4) => Attack(3)
b(3) AND c(1) AND d(3) => e(3)	Service (3) AND flag (1) AND src_bytes(3) => Attack(3)
b(3) AND c(1) AND d(4) => e(3)	Service (3) AND flag (1) AND src_bytes(4) => Attack(3)

Rules derivations need to be analyzed to ensure rules significance employed in classification process. In IDS, huge numbers of rules are generated in huge dataset. The concern of rule measurement is to determine only significant rules are utilized in classification process to guarantee better accuracy classification performance. Other researches on classification studied rules generation (Azevedo & Jorge, 2007; Carvalho et al., 2005; Lin, Ying, Lee, & Lee, 2012; Sengupta et al., 2013; Singh et al., 2013; Srikant & Agrawal, 1995; Ye, Yang, Geng, Zhou, & Chen, 2002) and focused on rules analysis. However, none of the studies research on comparative accuracy of classification performance in determining rule significance for the classifier. In this research, four types of rules measurement were employed which are support, accuracy, coverage and rule important measure.

3.4.4.1 Support measurement

Given a description of conditional part α and the decision part β; thus, the decision rule is $\alpha \rightarrow \beta$. Support of α is a number of objects in the information system A that has the property described by α (Li, 2007)

$$Support(\alpha) = ||\alpha|| \tag{3.2}$$

Support of β is number of objects in information system A that have decision described by β (Li, 2007).

3.3

Support for decision rule $\alpha \rightarrow \beta$ is the probability of object covered by the description belongs to the class (Li, 2007).

$$Support(\alpha \rightarrow \beta) = Support(\alpha \,.\, \beta) \tag{3.4}$$

3.4.4.2 Accuracy measurement

Quantity accuracy $(\alpha \rightarrow \beta)$ gives a measure of how trustworthy a rule is in condition β. It is the probability that an arbitrary object is covered by the description belongs to the class. It is identical to the value of rough membership function applied to an object x that match α. Thus, accuracy measures the degree of membership of x in X using attribute B (Li, 2007).

$$\text{Accuracy }(\alpha \rightarrow \beta) = \frac{Support(\alpha \,.\, \beta)}{Support(\alpha)} \tag{3.5}$$

3.4.4.3 Coverage measurement

The process measures the behavior of pattern α in describing decision class defined through β. It is a probability an arbitrary object belongs to C class and is covered by the description D (Li, 2007).

$$\text{Coverage } (\alpha \rightarrow \beta) = \frac{Support(\alpha \,.\, \beta)}{Support(\beta)} \tag{3.6}$$

It has to be noted that the rules are completed when any object which belongs to class is covered by description coverage.

3.4.4.4 Rule Importance Measure

Rules are generated from reducts, which are extracted from the dataset. It is worth noting that the rule sets, which are generated from different reducts are comprised of different sets of rules due to the fact that reduct is not unique. In most rule sets, important rules appear more than the less important rules. Rule Importance Measure (RIM) is computed depending on the recurrence of association rule among the rule sets (Li, 2007).

$$\text{Rule Importance Measure} = \frac{\text{Number of times a rule appears in all the generated rules from the reduct set}}{\text{Number of reducts set}} \tag{3.7}$$

Training dataset D as an input is labeled with duration = A, protocol type = B, service = C until attack attribute as shown in Figure 3.5. As an output from line 2 to 9, significance ruleset R is gathered for each class C that covers instances in D. In line 2 to line 5, R is the empty set and dataset D is nonempty set. In line 6 to 7, rule r is constructed to classify instances in dataset D which belong to class C. In line 8 to 9, rule r constructed is added to ruleset R and instances classified correctly by rule r is removed from dataset D. In line 10-13, support, accuracy, coverage and RIM are calculated and return to ruleset *R*. The enhancement strategy is proposed in line 1.6-1.9 to encourage better IDS performance.

```
Input: label training dataset $D$
Output: ruleset $R$ that cover instance in $D$
1    Start:
2        Set $R = \{Ø\}$
3        Set $N = C_{1...}C_n$
4            For each class $C$
5                While $D$ is nonempty
6                Construct one rule $r$
7                Classifies instance in D which belongs to class C
8            End for
         Add rule $r$ to ruleset $R$
9        Remove from $D$ instance classified correctly by $r$

10   Calculate Support ($T$) = ($Support(\alpha . \beta)$)
11   Calculate Accuracy ($U$) = ($Support(\alpha . \beta)$ / $Support(\alpha)$)
12   Calculate Coverage ($V$) = ($Support(\alpha . \beta)$ / $Support(\beta)$)
13   Calculate RIM ($W$) = (Number of appears reduct ($e$) / total reduct ($f$)

14   Return R
15   End
```

Figure 3.5 Significance Rule Algorithm

3.5 Measurement / Evaluation Criterion

Research tools used in the study are Rough Set Tool for Data Analysis (ROSETTA). It is RST experiment tool that is capable to generate reduct, rule and classification stage. Apart from that, Receiver Operating Characteristic (ROC) is an evaluation tool to validate the results obtained from the experiments. Both tools are further explained in next subchapters.

3.5.1 Receiver Operating Characteristic (ROC) Curve

Diagnostic performance test or test of accuracy is evaluated using "Receiver Operating Characteristic (ROC) curve analysis" (Metz, 1978; Zweig & Campbell, 1993). ROC curves are capable to distinguish diagnostic performances of diagnostic tests (Griner et al., 1981). Graph in Figure 3.6 shows three ROC curves indicator representing excellent, good, and worthless of true positive rate vs false positive rate performance tests plotted on the same graph. This graph is mapped to performance result obtained from Chapter 4 for validation and comparison. Test accuracy depends on how well the test separates the group with or without the attack state. ROC measures specific area to produce accurate performance.

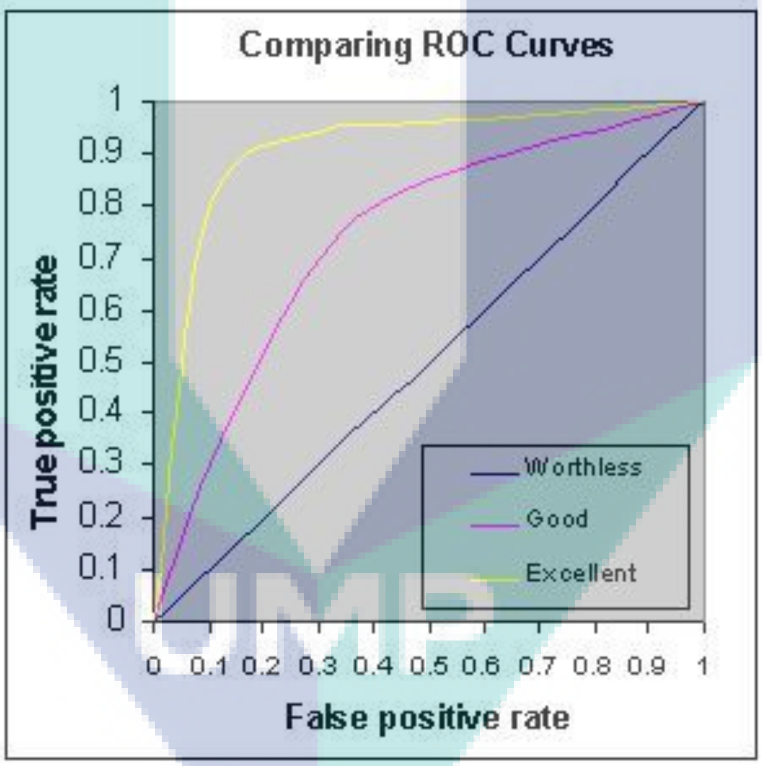

Figure 3.6 ROC Curve Sample
Source: Medcalc (2018)

3.6 IDS Evaluation Parameter

IDS is evaluated depending on several parameters such as accuracy, speed, detection, false positives, true negative etc. Unfortunately, nowadays IDS false alarms and accuracy are crucial issues need to be considered in designing efficient IDS (Sabri et al., 2011). Efficient IDS is able to determine correct event classification either attack or normal behavior using prediction (Wu & Banzhaf, 2010). As indicated by genuine way of a given occasion and forecast from IDS shown in Table 3.7 four possible outcomes are identified (false positive, false negative, true positive and true negative) known as confusion matrix (Wu & Banzhaf, 2010; Wu & Yen, 2009).

Table 3.7 Confusion Matrix

	Normal	**Attack**
Normal	True negative (TN)	False positive (FP)
Attack	False negative (FN)	True positive (TP)

Source: Wu & Banzhaf (2010); Wu & Yen (2009)

IDS need to evaluate the performance to classify event correctly as normal or attack. Accuracy is a fraction of test observations classified to the correct class (error rate = 1-accuracy). Unfortunately, accuracy may consist of insufficient information when the class contains different number of examples or errors are severe than making another. Given two classes of positive (false positive and true positive) and negative (false negative and true negative) observations;

i) False positives (FP) / false alarms are negative observations classified to positive class.
ii) False negatives (FN) are positive observations classified to negative class.
iii) True positives (TP)/detection rate are correctly classified as positive observations.
iv) True negatives (TN) are correctly classified as negative observations.

True negatives and true positives correlate to correct IDS operation; “True Negatives” (TN) events, which are actually normal, are labeled as normal, “True Positives” (TP) are events, which are actually attacks, and they are labeled as attacks. “False Positives” (FP) are normal event classified as attack; “False Negatives” (FN) are attack event incorrectly classified as normal event. Confusion matrix in Table 3.5

confirms that equation 3.8 – 3.13 shows numerical parameters applied following the measures to evaluate IDS performance. In validating IDS performance, parameter in confusion matrix as True Negative (TN), False Positive (FP), False Negative (FN) and True Positive (TP) are involved in the calculation.

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN} \qquad 3.8$$

$$\text{False Negative Rate (FNR)} = \frac{FP}{TP + FN} \qquad 3.9$$

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN} \qquad 3.10$$

$$\text{True Negative Rate (TNR)} = \frac{TN}{TN + FP} \qquad 3.11$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \qquad 3.12$$

$$\text{Precision} = \frac{TP}{TP + FP} \qquad 3.13$$

False Positive Rate (FPR) alludes that normal data is distinguished as an attack event. High FPR causes low IDS performance; meanwhile, high FNR exposes the system to intrusions due to its vulnerability. TNR also known as detection rate or sensitivity refers to detected attacks among attack events. Accuracy refers to event classified as accurate type in total events (Wu & Yen, 2009). Hence, effective IDS have to decrease FP and FN rates and maximize accuracy, TP and TN rates.

3.7 Experimental Set-Up on IDS Environment

Research tools used in the study are Rough Set Tool for Data Analysis (ROSETTA). RST experiment tool is capable to generate reduct, rule and classification stage, which is further explained in next subchapter. Apart from that, details in parameter selection and IDS evaluation parameters are also elaborated.

3.7.1 ROSETTA Tool

Rough Set Tool for Data Analysis or ROSETTA software tool for data mining applies Rough Set Theory approach as experiment tool used in this research. This toolkit is robust, user friendly and a powerful system based on discernibility data mining and knowledge discovery which is able to analyze tabular medical data (Ohrn, 1999). The system also exhibits originality as it is designed to cover all stages of knowledge discovery process.

3.7.2 Parameter Selection

In research experiments, several parameter selections are performed. In Reducer with Genetic Algorithm, which generates reduct, full discernibility is selected as it produces minimal attribute subsets. Modulo decision is also selected which does not discern similar generalized decision class. It is not necessary to have discernibility object in class for classification purpose. Other parameters and values are employed as shown in Table 3.8. Crossover probability, mutation probability and inversion probability values are set below than 1 because the machine is unable to compute the reduct due to machine specification performance. Similar reason also explains the values of crossover number points, individual number mutation and inversion number transportation which are set to 1 as minimum value depends on machine’s ability to process the reduct.

Table 3.8 Parameter and Value of Genetic Algorithm of Standard RS Reducer

Parameter	**Value**
Crossover Probability	0.3
Mutation Probability	0.05
Inversion Probability	0.05
Number of Crossover Points	1
Number Mutation on an Individual	1
Number Transposition for Inversion	1

3.7.3 Benchmark Dataset

KDD Cup 99 dataset is employed as benchmark dataset of IDS (detail in section 2.11 in Chapter 2). Continued use of KDD Cup 99 Data in current research report from Columbia University (Elkan, 2000; Wenke, 2000; Levin, 2000; Pfahringer, 2000) confirms the uniqueness of these dataset in offering large volume of network audit data with wide variety of labeled intrusions. For these reasons, researchers decided to use KDD Cup 99 dataset for the investigation of this research. KDD Cup 99 dataset is not only the most widely used dataset in intrusion detection, but is also the de facto benchmark on evaluating merits performance of intrusion detection system (Wang et al, 2014). KDD dataset is a well-known benchmark in the research of Intrusion Detection techniques (Agarwal & Sharma, 2015) and as explained by Mansoori et al. (2017), KDD Cup 99 data is a very popular and widely used intrusion attack dataset. Dataset is attached in Appendix A.

3.8 Conclusion

In summary, this chapter elaborates research methodology involved in constructing research output focusing on the enhancement of classification techniques, specifically in discretization algorithm and strategy of significance rule. The propose enhancements are evaluated against IDS benchmark dataset and validated using IDS parameter and ROC tool to improve IDS performance of accuracy, detection rate and false alarm. Generally, IDS requires high detection and low false alert rate to perform effectively.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Overview

In this study, KDD Cup 99 dataset consists of 4,898,431 and 311,029 records in training and testing set. Thus, 20,996 records data were used in training data, while 8,999 records as testing data with 41 condition attributes and one decision attribute are considered in the evaluation phase. The proposed Frequency Binning discretization and a new strategy to generate significance rule is discussed in Chapter 3. It is implemented into RST to be tested on IDS application with the mentioned data, to evaluate the performance of accuracy, detection and false alarm rate. The rest of this chapter focuses on explaining and analysing the obtained result from the executed experiment to relate to the listed objectives.

Threat of validity in this experiment is networked through data nature. It should not prevent IDS from executing attack detection and should not influence the accuracy and rate detection of attacks. Other threat to this validity is the amount of time IDS takes to identify an intrusion attempt, also new attack is excluded from the benchmark data.

4.2 Result of Proposed Frequency Binning Discretization

In order to evaluate the proposed discretization algorithm, available discretization algorithm in RST named as Equal Frequency Binning (EFB), Naives, Entropy/MDL were implemented and evaluated. Table 4.1 show the results of proposed algorithm compared to the mentioned algorithm. Based on EFB algorithm, there are 17 bins allocated to the value of numeric numbers (refer to Appendix B), with 98.21% accuracy performance. The values of bin allocated and produced by Entropy/MDL is 27 bins with 98.03% accuracy performance, and Naives algorithm generates 35 bins with 96.968% accuracy performance. On the other hand, the proposed Frequency Binning shows that a number of bins equal to 3 yields 98.31% classification accuracy. Meanwhile, a large number of bins favors low classification accuracy. Naive algorithm possesses bin number of 38 with 96.96% of classification accuracy. Thus, the proposed work shows the best result with 3 bins and 98.31% accuracy.

The experiment indicates that those who have few bins, their classification rates increase because the lesser bin generates data map precisely. All related values are mapped into the correct bin. As more bins results in waste bin (bin with no data) and some data is not taken to bin due to certain value does not belong accurately to the generated bins. On the other hand, discretization process is to decrease the amount number of attribute values to manageable amount of attribute values. Discretization is conducted by dividing the range of attributes values into *k* intervals with equal width to have equal size of bin. In other common discretization algorithms, EFB does not use class labels, the classification information possibly loses different classes can easily grouped together and sensitive n outlier.

In EFB, the interval bin consists the same range of numerical values. Each range is generated automatically. Consider the case which there is a range value between 1 and 20, and the value of 100 and *k* = 5. Consequently, EFB is generating 15 empty bins, however the discretization resulting a meaningless attributes distribution. Therefore, the Proposed Frequency Binning Discretization algorithm adopts fewer numbers of bins at equals to 3 for reduct generation and rules derivation. The minimum number of bins depends on how much dataset is involved in the classification. The proposed algorithm generates the minimum bin suits to amount of employed attributes object.

Table 4.1 IDS Classification Percentage Using Discretization Algorithm

Discretization Algorithm	Bin Number	% Classification
Equal Frequency Binning	17	98.21
Entropy/MDL	27	98.03
Naive	35	96.96
Proposed Binning	3	98.31

From the discretization experiments, it depicts the effect of accuracy percentage of classification when the algorithm varies with different sizes of bins. It indicates that the minimum value of 3 bins among other discretizer (refer to Table 4.1) is convincing in order to produce high impact of accuracy rate in these instances. The minimum number of intervals is considered the optimal number of bin in the training data. The $q^a \ldots q^n$ is precisely taken the values to number of bin, starting with first value map until end of value. The proposed discretization algorithm does not stick to a minimum value of 3 bins The minimum number of bin depends on how much dataset is involved in the classification. The lesser bin, the less time consuming taken in the processing, and the higher accuracy performance rate gathered, however the time processing is not involved in the scope of research.

4.3 Result of Proposed Strategy of Significance Rule

From the experiments result, significance attributes were generated as shown in Table 4.2. Originally, there are 42 attributes including decision attributes. However, only 31 core attributes are revealed through the simplification decision table analysis. The reason is the core attributes are selected based on feature selection process conducted in reduct stage, with two parameters, core attributes and minimal cardinality. The core attribute examines the indiscernibility object attributes means unique attributes are searched to be concrete attributes employed in classification process to gain better accuracy performance. Meanwhile minimal cardinality is a minimal set of object value in attributes is searched. Since discretization converts continuous values into discrete values that have finite cardinality, rather than real or continuous values have an infinite cardinality.

116 reducts (refer to Appendix C) were derived from standard RST reducer with support of 100 and cardinality length of 2, 3 and 4. In the experiment, the reduct with minimal cardinality with length of 2, meant that only two attributes in the set, were considered as the significant attribute because of minimal cardinality specify less attribute reduct, in core subset of attributes. Refer to Appendix D for the list of generated rules in standard RST classification.

Simplification decision tables were mapped, refer to the employed attributes in Table 4.2, based on significance reduct of core attributes and attributes with minimal cardinality in the proposed classification technique for IDS based on RST. Refer to Appendix E for the list of generated reduct in simplification decision table. In RST rule generation output (refer to Appendix F), the accuracy and coverage with value 1.0 are the significance value rather than value below 1.0. The coverage of attribute usage value, min support value, mean support value and max support value are favor in the proposed classification technique, since those values are involved in accuracy calculation generated.

Table 4.2 Simplification Decision Table of KDD Cup 99

No	Attributes	No	Attributes	No	Attributes	No	Attributes
1	duration	9	num_ compromised	17	srv_count	25	dst_host_same_ srv_ rate
2	protocol_ type	10	root_shell	18	serror_rate	26	dst_host_diff_ srv_ rate
3	service	11	num_root	19	srv_rerror_rate	27	dst_host_same_ src_ port_rate
4	flag	12	num_file_ creations	20	same_srv_rate	28	dst_host_srv_ diff_ host_rate
5	src_bytes	13	num_shells	21	diff_srv_rate	29	dst_host_srv_ serror_ rate
6	dst_bytes	14	num_access_ files	22	srv_diff_host_ rate	30	dst_host_rerror_ rate
7	hot	15	is_guest_login	23	dst_host_count	31	type_attack
8	logged_in	16	count	24	dst_host_srv_ count		

Total of 173 rules with number of support, coverage and accuracy values with length 2,3,4 and 5 has been generated, after the process of reduct with standard RST reducer from the phase A work (details in Chapter 3). Thorough rules are analyzed by comparing the highest LHS and RHS support values, searching the accuracy with value 1.0 and also involving the comparing the minimal cardinality of LHS and RHS length among of all rules. There is 1 rule with highest support which is;

dst_bytes([279, 1695)) AND hot([*, 1)) => type_attack(normal.) with 7967 LHS and RHS support value, RHS accuracy is 1.0 and LHS and RHS length are 2 and 1 in each.

The value of 7967 LHS and RHS is the highest support value, meaning that the rule is with the highest confidence in the rules generated, with high accuracy of 1.0 with least length cardinality which is significance rule to be employed in classification process the to deliver high accuracy classification performance.

Rules with coverage 1.0 which are the fraction of the records that satisfied the "IF and THEN" conditions of the rules are;

i) service(ecr_i) AND src_bytes([358, *)) => type_attack(smurf.) with RHS and LHS support values 4971
ii) num_compromised([2, *)) AND num_file_creations([1, 2)) => type_attack(buffer_overflow.)with RHS and LHS support values 3.
iii) root_shell(1) AND num_access_files([1, 2)) => type_attack(phf.) with support values 1.

The coverage 1.0 is the highest value instead of below 1.0, meaning that the most often attributes in the rules generated are service, src_bytes, num_compromised, num_file_creations, root_shell and num_access_files which are significantrules to be employed in classification process to gain high accuracy performance. Refer to Appendix G for list of generated rules with coverage of value 1.0.

Table 4.3 shows 9 rules with minimal length of 1 and 2 as the number of conditional elements in IF part, with different support value, LHS and RHS coverage with accuracy 1.0 (high accuracy compared to <1.0 value). Refer to Appendix H for list of generated rules with accuracy of value 1.0 and Appendix I for the list of generated rules with length 2 in simplification decision table.

Table 4.3 Rules with Minimal Length of 1

No	Rules	Support	Accuracy	LHS Coverage	RHS Coverage	LHS Length	RHS Length
1	dst_bytes([279, 1695)) AND hot([*, 1)) => type_attack(normal.)	7967	1.0	0.332014	0.443869	2	1
2	duration([*, 1)) AND src_bytes([240, 358)) => type_attack(normal.)	7867	1.0	0.327846	0.438297	2	1
3	service(http) AND src_bytes([240, 358)) => type_attack(normal.)	7875	1.0	0.32818	0.438743	2	1
4	duration([*, 1)) AND dst_bytes([279, 1695)) => type_attack(normal.)	7353	1.0	0.306426	0.409661	2	1
5	service(http) AND dst_bytes([279, 1695)) => type_attack(normal.)	6681	1.0	0.278421	0.372221	2	1
6	dst_host_srv_count([255, *)) AND dst_host_same_src_port_rate([0.01, 0.09)) => type_attack(normal.)	6334	1.0	0.263961	0.352889	2	1
7	dst_bytes([279, 1695)) AND dst_host_srv_count([255, *)) => type_attack(normal.)	6313	1.0	0.263086	0.351719	2	1
8	protocol_type(tcp) AND src_bytes([*, 240)) AND hot([*, 1)) AND dst_host_srv_count([255, *)) => type_attack(normal.)	6095	1.0	0.254001	0.339573	4	1
9	protocol_type(tcp) AND dst_host_count([*, 81)) AND dst_host_srv_count([255, *)) => type_attack(normal.)	6043	1.0	0.251834	0.336676	1	1

Refer to Table 4.4, a different sampling of KDD Cup 99 dataset was employed to test the proposed classification technique. The experiments show several sample sizes with different objects (same total attributes number) contribute to different reduct, different rules with different accuracy classification performance.

Table 4.4 Different Size Sampling of KDD Cup 99 of Proposed Classification

Sample Size	Number of Objects	Number of Attributes	Number of Reducts	Number of Rules	% Classification
10 K	10,000	42	113	58,773	99.2
15 K	15,000	42	133	84,447	99.68
20 K	20,000	42	45	13,090	99.85
30 K	30,000	42	113	6,114	99.34

Furthermore, Table 4.5 shows different sampling frequency with different number of objects in different distribution of types attack that need to be classified.

Table 4.5 Frequency of Attack in Dataset Refer to Respective Attack Classess

Attack Class	Dataset Sample Size			
	10 K	15 K	20 K	30 K
Normal	7751	12748	12891	1209
DoS	2008	2011	6868	28526
Probe	232	232	232	70
R2L	8	8	8	195
U2R	1	1	1	0

4.3.1 Significance Rule with Highest Number of Support Value

The following rules are implemented to ensure their support value to reveal the most significant rules. The highest support value results in the most significant rules. The statistics of rules involved in this analysis are rule support, rule coverage and rule accuracy. Rules computation are conducted by sorted of highest rule support values, Refer to the definition of the highest support rules. The considered rules are listed as below;

i) dst_bytes([279, 1695)) AND num_file_creations([*, 1)) => type_attack(normal.) with LHS and RHS length values 2 and 1 correspondence

ii) dst_bytes([279, 1695)) AND num_file_creations([*, 1)) AND dst_host_srv_serror_rate(0) => type_attack(normal.) with LHS and RHS length values 3 and 1 correspondence

iii) dst_bytes([279, 1695)) AND num_compromised([*, 1)) AND num_file_creations([*, 1)) => type_attack(normal.) with LHS and RHS length values 3 and 1 correspondence

iv) protocol_type(tcp) AND dst_bytes([279, 1695)) AND num_file_creations([*, 1)) => type_attack(normal.) with LHS and RHS length values 3 and 1 correspondence

The rules are considered the most significant rules supported by the highest value of 8019 for LHS and RHS support value, and accuracy value 1.0 with the outcome of type_attack (normal). LHS support contains the number of records in training data that fully exhibit the attribute property. On the other hand, while RHS support contains the number of records in training data that fully exhibits the property, which is described by the THEN condition.

4.3.2 Significance Rule with Accuracy 1.0 and Coverage 1.0

The generated rules with accuracy values 1.0 consist of 12,458 from 12,471 in total, Refer to Table 4.6. There are different LHS and RHS support values with accuracy 1.0.

Table 4.6 Rule Accuracy Computations based on Rule Derivation

No	Rules	LHS Support	RHS Support	Accuracy Computation
1	src_bytes([240, 358)) AND count([4, 17)) => type_attack(normal.)	4125	4125	4125/4125=1.0
2	src_bytes([240, 358)) AND count([17, *)) => type_attack(normal.)	1549	1549	1549/1549=1.0
3	src_bytes([240, 358)) AND root_shell(0) => type_attack(normal.)	7957	7957	7957/7957=1.0
4	src_bytes([358, *)) AND dst_bytes([279, 1695)) => type_attack(normal.)	1435	1435	1435/1435=1.0
5	service(http) AND src_bytes([240, 358)) => type_attack(normal.	7875	7875	7875/7875=1.0

Table 4.7 consists of the rules with accuracy and coverage 1.0 with RHS and LHS support values. LHS and RHS supports value are sorted. The last rule which is logged_in(0) AND dst_host_srv_count([255, *)) => type_attack(smurf.) have the highest RHS and LHS support value with 4948. This rule contains the highest number of records which fully exhibit the attribute property.

Table 4.7 Rules with Accuracy and Coverage Value 1

No	Rules	RHS & LHS Support	Accuracy	Coverage
1	src_bytes([358, *)) AND hot([1, 3)) AND num_compromised([1, 2)) AND dst_host_diff_srv_rate(0) => type_attack(back.)	768	1.0	1.0
2	duration([*, 1)) AND num_compromised([1, 2)) AND dst_host_same_srv_rate(1) => type_attack(back.)	767	1.0	1.0
3	flag(SF) AND src_bytes([358, *)) AND num_compromised([1, 2)) AND num_access_files([*, 1)) => type_attack(back.)	759	1.0	1.0
4	duration([*, 1)) AND src_bytes([358, *)) AND hot([1, 3)) AND srv_diff_host_rate(0) => type_attack(back.)	764	1.0	1.0
5	src_bytes([358, *)) AND hot([1, 3)) AND num_root([*, 1)) AND serror_rate(0) => type_attack(back.)	792	1.0	1.0

Table 4.7 Continued

No	Rules	RHS & LHS Support	Accuracy	Coverage
6	protocol_type(icmp) AND srv_count([23, *)) AND dst_host_srv_count([255, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(smurf.)	4947	1.0	1.0
7	flag(SF) AND src_bytes([358, *)) AND logged_in(0) AND dst_host_srv_count([255, *)) => type_attack(smurf.)	4948	1.0	1.0

Rule Importance Measure (RIM) is utilized to assess the significance of association rules. As mentioned in Chapter 2, is defined as;

$$\text{Rule Importance Measure} = \frac{\text{Number of times a rule appears in all the generated rules from the reduct set}}{\text{Number of reducts set}} \qquad 4.1$$

The analysis of RIM is demonstrated to show the significance of selected attributes. The number of reduct set generated from simplification decision table is 8083 and the attributes {dst_host_srv_diff_host_rate} have the most noteworthy RIM percentage which is 49.58% as shown in Table 4.8.

Table 4.8 Rule Important Measure from Simplification Decision Table

Attribute	Calculation RIM	Total RIM
duration	1550/8083	19.18%
protocol_type	1157/8083	14.31%
service	3333/8083	41.23%
flag	951/8083	11.77%
src_bytes	3030/8083	37.48%
dst_bytes	2746/8083	33.97%
hot	1743/8083	21.56%
logged_in	1476/8083	18.26%
num_compromised	1111/8083	13.74%
root_shell	797/8083	9.86%
num_root	747/8083	9.24%

Table 4.8 Continued

Attribute	Calculation RIM	Total RIM
num_file_creations	787/8083	9.74%
num_access_file	839/8083	10.38%
1s _guest_login	427/8083	5.28%
count	2211/8083	27.35%
srv_count	1921/8083	23.77%
serror_rate	719/8083	8.90%
srv_rerror_rate	955/8083	11.81%
same_srv_rate	756/8083	9.35%
diff_srv_rate	832/8083	10.29%
srv_diff_host_rate	1416/8083	17.52%
dst_host_count	2715/8083	33.59%
dst_host_srv_count	1584/8083	19.60%
dst_host_same_srv_rate	1817/8083	22.48%
dst_host_diff_srv_rate	1489/8083	18.42%
dst_host_same_src_port_rate	2709/8083	33.51%
dst_host_srv_diff_host_rate	4008/8083	49.58%
dst_host_srv_serror_rate	704/8083	8.71%
dst_host_rerror_rate	979/8083	12.11%
dst_host_srv_rerror_rate	899/8083	11.12%

Therefore, all rules in {dst_host_srv_diff_host_rate} attributes are chosen for the classification process; thus, contributing to increase the classification accuracy. The 12,458 of significance rules are determined, from 12,471 rules in total. These significant rules are conceded for classification process to enhance the classification percentage.

High classification was accomplished with the proposed technique. IDS does not need to utilize all attributes and rules to diagnose the pattern of attacks, since the attack detection a long processing time, and no guarantee of better performance. Subsequently, the core attributes and the significant rule are favored for quick decision making in determining better results for classification. Table 4.9 outlines the percentage of classification performance of RST decision table and the RST simplification decision table of KDD Cup 99 dataset.

Table 4.9 Classification Performance of KDD Cup 99 Dataset

Classification	Rule set	Overall Accuracy
RST decision table	All rules	98.31%
RST simplification decision table	Selected rules	99.95%

Table 4.10 shows comparative research work analysis including the proposed work result. In research work of Junyuang, Jidong & Hao (2012) and Zhang et al. (2012), different classifier is used as Rough Set Theory, Artificial Immune System and Support Vector Machine. Different numbers of attributes are implemented as accuracy performance of classification is at 98.25% and 95% respectively. Hamid et al (2015) enhanced feature selection phase using RST. 21 attributes are selected with the result of 99.95% accuracy classification performance. Yang (2016) employed RST in IDS classification with 31 attributes resulting in 78.3% of accuracy performance of classification. All research works employed RST as the classifier and also as the preprocessing tools in IDS classification. Hamid et al (2015) rank the highest percentage, meanwhile Yang (2016) gains the lowest accuracy percentage. Hamid et al (2015) embark their percentage in the impact of RST feature selection, they do labeling for the feature, and enhance the preprocessing algorithm. Meanwhile, Yang (2016) only employed the standard RST into their research work. The proposed technique involving enhancement discretization algorithm and strategy of generating significance rules, contribute to 99.95% accuracy classification performance with 31 attributes.

Table 4.10 Comparison Between Proposed Classification and Other IDS Research Using Rough Set Theory

Author	Classifier	Year	Accuracy %	Feature Selection	No of Attribute
Junyuang, Jidong & Hao	• Improved Artificial Immune System • Rough Set Theory	2012	98.25	N/A	42
Zhang, Jia, Shi, Tang, & Wang	• Rough Set Theory • Support Vector Machine	2012	95	N/A	29

Table 4.10 Continued

Hamid H. Jebur , Mohd Aizaini Maarof , Anazida Zainal	• Enhancing Rough Set Theory	2015	99.95	Yes	21
Yang Hui Jun	• Rough Set Theory	2016	78.3	Yes	31
Proposed work	• Enhancing Frequency Binning Discretization and Enhancing Strategy of Significance Rule	2018	99.95	Yes	31

Table 4.11 shows the classification results of four attack categories which are Probe, DoS, U2R and R2L. Normal class is the secure state; meanwhile the undefined class is the unclassified state. The proposed technique shows the enhanced algorithm and strategies that were able to generalize in detecting those attacks and normal decision. Basic RST classification generates the same number of detection between RST decision table and RST simplification decision table. However, normal and undefined classes are in different values. RST classification detects normal state in 4399 and 101 undefined state, meanwhile proposed RST classification detect 4497 normal state and 3 undefined state. Different values of normal and undefined state show that the proposed RST classification is able to detect 4500 objects of (4399 + 101) = (4497 + 3) normal and undefined. As a result, it decreases the number of object in undefined class. Undefined class is a harm situation where IDS label the object value as unclassified state which should be discarded in IDS database; thus, the users have safe online activity in Internet.

Table 4.11 Number of Detection Between RST Decision Table and RST Simplification Decision Table

Class Attack	RST classification	Proposed RST classification
Probe	54	54
R2l	1	1
Dos	1444	1444
U2r	0	0
Normal	4399	4497
Undefined	101	3

To diagnose attack patterns were detected. All the attributes and rules were not utilized in IDS since all attributes and rules involved will result in longer processing time of classification process and there is no guarantee of high performance. In the training and testing section, discretization is an important stage as it can contribute to better accuracy. Thus, for quick decision making in increasing high result for classification, the core attributes and the significant rules are favored. Noteworthy, impact on the classification rates yields by new generation of decision table for KDD Cup 99 dataset. Significance attributes and rules are yielded by the reduct and rules analysis. The significance attributes and rules are proven to increase classification accuracy compared to the results yielded from the employment of all attributes, reduct and generated rules. Table 4.12 demonstrates comparative analysis of true rate, accuracy rate, false and true positive rate.

Table 4.12 IDS Evaluation Parameter of Accuracy Performance

IDS Evaluation Parameter	Percentage
True Rate	99.95%
Accuracy Rate	98.31%
False Positive Rate	0.0473%
True Positive Rate	1.6836%

Table 4.13 and 4.14 elaborate on the accuracy and false positive alarm rates comparison between RST decision table. Proposed simplification decision table, Decision Tree ID3 and Decision tree and Stratified weighted Sampling. Decision Tree constructs process of top down, divide and conquers and greedy algorithm. Meanwhile, ID3 algorithm is good to limited dataaset, however, ID3 is not capable in handling missing values. Furthermore when size of dataset is increased, the tree is not

accustomed to the changes (Ming et al., 2009). Stratified Weighted Sampling produce the samples from dataset, then the decision tree algorithm is applied. This algorithm is designed to encounter the ID3 limitations (Devendra & Jain, 2012). The proposed RST seems to generate the encouraging result in accuracy and false positive rate of four different samples (10K, 15K, 20K, 30K) in IDS dataset, compared to RST, ID3 and Decision Tree and Stratified Weighted Sampling. Proposed work had shown the capability in handling missing value, precise bin of discretization value and significance rules in IDS classification. Figure 4.1 graphically shows that proposed work is in higher accuracy percentage in those four different samples classification with 31 features compared to 42 features classification.

Table 4.13 Comparison of Accuracy Rate

Algorithm	**Dataset**			
	10 K	**15 K**	**20 K**	**30 K**
Basic RST	99.2%	99.68%	99.85%	99.34%
ID3	88.78	88.23%	87.86%	86.45%
Decision Tree and Stratified Weighted Sampling	94.74%	94.54%	94.02%	93.85%
Proposed RST	99.6%	99.86%	99.95%	99.46%

Table 4.14 Comparison of False Positive Rate

Algorithm	**No of features**	**Dataset**			
		10 K	**15 K**	**20 K**	**30 K**
Basic RST	42	0.8%	0.31%	0.15%	0.2%
ID3	41	8.78%	8.24%	9.64%	9.81%
Decision tree and Stratified Weighted Sampling	41	2.81%	3.26%	3.18%	3.92%
Proposed RST	31	0.4%	0.13%	0.05%	0.53%

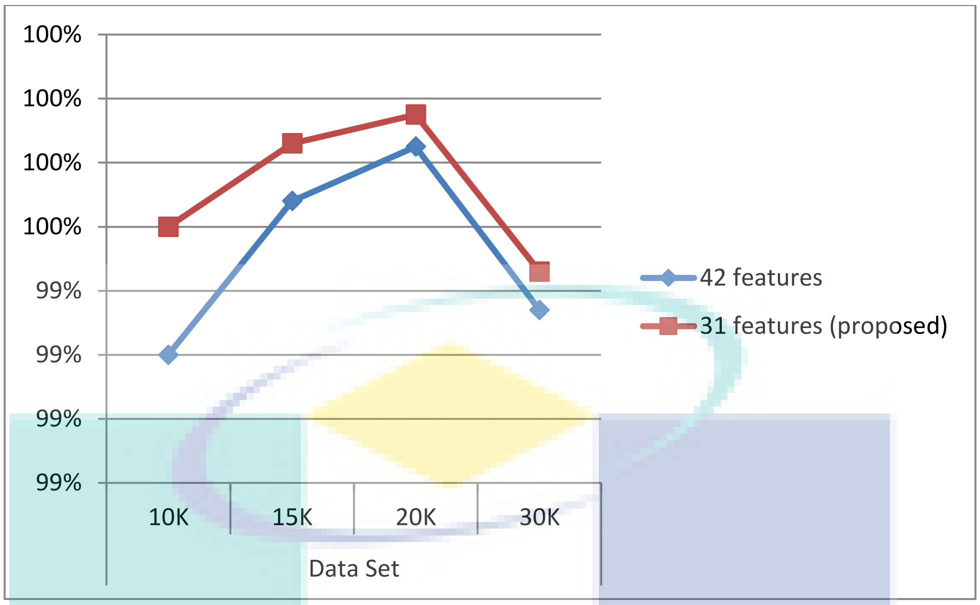

Figure 4.1 Analysis of Classification Accuracy Rate of IDS Using Two Features Set

Area Under Curve (AUC) of ROC Curve measure sthe efficiency for the classifier in the mining task. ROC curve illustrates the performance of binary classifier system. Multiclass classification problem treats binary classifier as one versus all and calculates the operating point for each class, the take out result by calculating the average of all. Results of proposed IDS classification are; (shown in Figure 4.2)

Area under curve = 0.5503

Refer to AUC curve for accuracy classification using proposed model, the evaluation result is validated based on the worthless, good and excellent threshold mapping to threshold cutoff. The AUC value is measured within the value of 0 to 1; which means 1 (yellow line) is the highest performance accuracy (excellent), the value of 0.5 is good performance (red line) and 0 is the worthless value of accuracy performance (blue line). If AUC value is below 0.5, this value indicates that the proposed method needs to be remodel. The proposed technique is with the AUC curve of 0.5503 value (green line), means that the proposed technique is validated in above good accuracy performance.

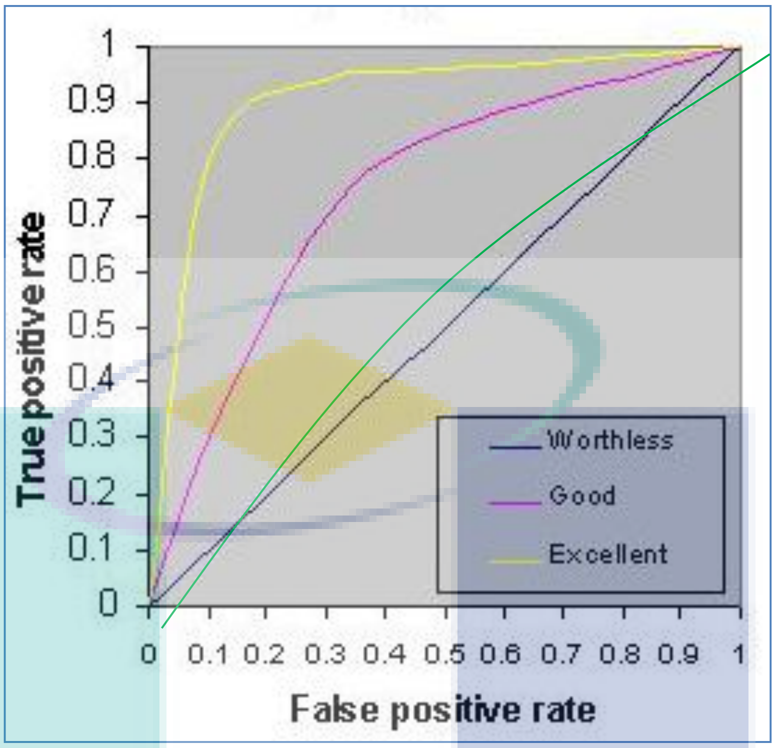

Figure 4.2 Proposed Classification of AUC Curve

CHAPTER 5

CONCLUSION

5.1 Overview

This chapter highlights the main contributions of study and implication for society and research community. This research focuses on improving classification technique based on Rough Set Theory to increase the accuracy and detection rate of IDS performance in detecting attacks.

5.2 Contribution

This research effort was aimed to enhance classification technique based on Rough Set Theory for Intrusion Detection System application, in term of accuracy, rate of detection and reducing false positive alarm rate. In order to achieve this aim, three objectives have been determined. Concerning the first objective, this research succeeds in proposing the enhancement of discretization based on binning discretization. This proposed technique has been tested, evaluated and compared to the other research work. The result shows the proposed work is able to achieved 98.31 % of accuracy on performance of IDS application.

Addressing the second objective, proposed technique in the first objective was integrated to the RST classification to improve classification performance. The result of accuracy and rate detection against IDS benchmark dataset is encouraging compared to other related research. The main key aspect of first and second objective are satisfied owing to the successful implementation of IDS classification.

As for the final objective, classification accuracy of finding of experiment is validated through IDS validation parameter using ROC tool to evaluate the findings. The evaluation contributes the proven validated output based on benchmark of IDS validation. An enhancement of classification technique against IDS application has been successfully employed to undertake all experimentations. Hence, it highlights the accuracy and detection for attack classification. In the conducted evaluation, proposed classification results are successfully compared against other Rough Set Theory model. Enhancement classification technique of IDS application experimental results is encouraging as many IDS classification researches are introduced.

Summing up, based on earlier discussion, main contribution of this work relates to new discretization algorithm and new strategy of employed in IDS classification. The research contribution undertaken in this research work is an enhancement of IDS application classification based on RST involving the enhancement of IDS algorithm of Frequency Binning for RST discretization process and enhancement strategy to generate significant rules for RST rules process.

5.3 Limitation of Work

The real network traffic dataset including data nature and network throughput is not involved in the research study. The time of computation processing is also not in the scope of research to be covered.

Regarding the growth of attack, new attack does not involve in IDS dataset. As an IDS state-of-the-practice benchmark dataset, the dataset does not consist a real time data to be classified to proposed enhancement technique.

5.4 Future Work

This thesis has proposed classification technique based on RST against IDS application. Some limitations were identified as discussed in previous section. Hence, to overcome the limitation and improve current work, the following ideas were suggested as future works;

i) To address the issues of classification performance in current proposed solution, the proposed algorithm can be made more efficient by incorporating hybrid classifier to gain better attack classification performance. The proposed algorithm and strategy can be extended by adapting other significant parameters of unit classifier. With the expanding episodes of cyberattacks classification, building powerful classification technique of interruption detection algorithms with high accuracy and real time performance are needed.

ii) KDD Cup 99 dataset is used in this research evaluation because it is relevant to be shared with other researchers to enhance the results. Nowadays, IDS research focuses on real time network performance. The challenge is to evaluate and enhance previous research result. In future work, proposed classification technique can be tested on other different IDS dataset for analyzing the capability of the proposed technique in improving attack classification performance.

REFERENCES

Aburomman, A. A., & Reaz, M. B. I. (2013). Evolution of Intrusion Detection System Based on Machine Learning Methods. *Australian Journal of Basic and Applied Sciences*, *7*(7), 799-813.

Aggarwal, P., & Sharma, S. K. (2015). An Empirical Comparison of Classifiers to Analyze Intrusion Detection. *Procceding of the 5th International Conference on Advanced Computing & Communication Technologies 2015, IEEE*, India, 446-450.

Agrawal, R., & Srikant, R. (1994). Fast Algorithms For Mining Association Rules. *20th Procceding of the International Conference on Very Large Databases*, Chile, 487–499.

Ahmad, P., Qamar, S., & Rizvi, S. Q. A. (2015). Techniques of Data Mining in Healthcare: A Review. *International Journal of Computer Applications*, *120* (15), 38-50.

Akbar, S., Rao, D. K. N., & Chandula, J. A. (2010). Intrusion Detection System Methodologies Based on Data Analysis. *International Journal of Computer Application*, *5*(2), 10-20.

Allen, J. (1990). Boolean Analysis. *Boolean Reasoning* (pp. 87-122). Retrieved from http://www2.fiit.stuba.sk/~kvasnicka/Free%20books/Brown_Boolean%20Reasoning.pdf (last accessed January, 2013).

Altwaijry, H., & Algarny, S. (2012). Bayesian based Intrusion Detection System. *Journal of King Saud University - Computer and Information Sciences*, *24*(1), 1-6.

Anderson, J. (1980). *Computer Security Threat Monitoring and Surveillance*. (Report No 79F296400). Fort Washington: James P. Anderson Co.

Atilla, O., & Hamit, E. (2016). A Review of KDD99 Dataset Usage in Intrusion Detection and Machine Learning between 2010 and 2015. *Journal of PeerJ PrePrints*, *4*, 1–21.

Azevedo, P. J., & Jorge, M. (2007). Comparing Rule Measures for Predictive Association Rules. *Procceding of the 18th Conference of European on Machine Learning*, Poland, 510-517.

Bahrainian, S., & Dangel, A. (2013). Sentiment Analysis uses Sentiment Features. Lecture Notes in Computer Science, Vol: 4701. *Procceding of the International Joint Conference of Web Intelligence and Intelligent Agent Technologies* (pp. 26–29).

Biswanath, M., Todd , L. H., & Karl N. L. (1994). Network Intrusion Detection. *IEEE Network*, *8*(3), 26–41.

Borgohain, R. (2012). FuGeIDS: Fuzzy Genetic Paradigms in Intrusion Detection Systems. *International Journal of Advanced Networking and Applications*, *3*(6), 1409-1415.

Bose, I. (2006). Deciding the Financial Health of Dot-Coms using Rough Sets. *Journal of Information and Management*, *43*(7), 835-846.

Bruha, I. (1997). Quality of Decision Rules: Definitions and Classification Schemes for Multiple Rules. *Machine Learning and Statistics: The Interface* (pp. 107–131). New York : John Wiley.

Carvalho, D. R., Freitas, A. A., & Ebecken, N. (2005). Evaluating the Correlation Between Objective Rule Interestingness Measures and Real Human Interest. Lecture Notes in Computer Science, Vol: 3721. *Procceding of the Conference of European on Principles of Data Mining and Knowledge Discovery* (pp. 453-461).

Catania, C. A., Bromberg, F., & Garino, C. G. (2012). An Autonomous Labeling Approach to Support Vector Machines Algorithms for Network Traffic Anomaly Detection. *Expert Systems with Applications*, *39*(2), 1822-1829.

Catlett, J. (1991). On Changing Continuous Attributes into Ordered Discrete Attributes. Lecture Notes in Computer Science, Vol: 482. *Procceding of the European Working Session on Learning* (pp. 164-168).

Chen, X., Vorvoreanu, M., & Madhavan, K. (2014). Mining Social Media Data to Understand Student's Learning Experiences. *IEEE Transaction on Learning Technologies*, *7*(3), 246–259.

Chi, C., Wee, P. T., & Guang, B. H. (2012). Extreme Learning Machines for Intrusion Detection. *Procceding of the 2012 International Joint Conference on Neural Networks*, USA, 1-8.

Dai, J. H., & Li, Y. X. (2002). Study on Discretization based on Rough Set Theory. *1st Procceding of the International Conference on Machine Learning and Cybernetics*, China, 3, 1371-1373.

Debar, H. (2009). An Introduction to Intrusion Detection Systems. *IBM Research, Zurich Research Laboratory*, 1-18.

Desale, K. S., & Ade, R. (2015). Genetic Algorithm based Feature Selection Approach for Effective Intrusion Detection System. *Procceding of the 3rd International Conference of Computer Communication and Informatics*, India, 1-6.

Devendra, K., & Jain, R. C. (2012). Improve Intrusion Detection using Decision Tree with Sampling. *International Journal of Computer Technology & Applications*, *3*(3), 1209–1216.

Dhakar, M., & Tiwari, A. (2014). A Novel Data Mining based Hybrid Intrusion Detection Framework. *Journal of Information and Computing Science*, *9*(1), 037–048.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and Unsupervised Discretization of Continuous Features. *Procceding of the 12th International Conference on Machine Learning*, USA, 194–202.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and Unsupervised Discretization of continuous Features. *Procceding of the 12th International Conference on Machine Learning*, USA, 194–202.

Elfeshawy, N. A., & Faragallah, O. S. (2012). Divided Two Part Adaptive Intrusion Detection System. *Springer Science+Business Media*, 301-321.

Garcia, S., Luengo, J., Saez,J., Lopez,V., & Herrera, F. (2013). Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning. *IEEE Transactions on Knowledge and Data Engineering*, *25*(4), 734-750.

Gautam, G., & Yadav, D. (2014). Sentiment Analysis of Twitter Data using Machine Learning Approaches and Semantic Analysis. *Procceding of the 7th International Conference on Contemporary Computing*, India, 437–442.

Gera, M., & Goel, S. (2015). Data Mining - Techniques, Methods and Algorithms: A Review on Tools and their Validity. *International Journal of Computer Applications*, *113*(18), 22-29.

Ghorbani, A. A., Lu, W., & Tavallaee, M. (2010). Network Intrusion Detection and Prevention Concepts and Techniques. *Advances in Information Security, Springer*.

Gokulakrishnan, B., Plavnathan, P.,Thiruchittampalam, R., Prasat, N., & Ashehan, P. (2012). Opinion Mining and Sentiment Analysis on a Twitter Data Stream. *Procceding of the International Conference on Advances in ICT for Engineering Regions*, Sri Lanka, *7*(3), 182–188.

Govindarajan, M., & Chandrasekaran, R. M. (2011). Intrusion Detection Using Neural Based Hybrid Classification Methods. *International Journal of Computer and Telecommunications Networking*, *55*(8), 1662-1671.

Griner, P. F., Mayewski, R. J., Mushlin, A. I. & Greenland, P. (1981). Selection And Interpretation of Diagnostic Tests and Procedures. *Annals of Internal Medicine*, *94*(4), 557–-592.

Guoyong, W. (2001). Rough Sets Theory and Knowledge Acquisition. *Xi'an Jiaotong University Press.*

Hacibeyoglu, M., Arslan, A., & Kahramanli, S. (2011). Improving Classification Accuracy with Discretization on Datasets Including Continuous Valued Features. *International Journal of Computer and Information Engineering, 5*(6), 623–626.

Hamid, Y., Sugumaran, M., & Journaux, L. (2016). Machine Learning Techniques for Intrusion Detection: A Comparative Analysis. *International Journal of Computer Application*, 1-6.

Han, J., Kamber, M. (2006). Datamining: Concepts and Techniques. *Morgan Kaufmann*. 3rd Edition.

Hassan, M. M. M. (2013). Network Intrusion Detection System using Genetic Algorithm and Fuzzy Logic. *International Journal of Innovative Research in Computer and Communication Engineering*, *1*(7), 1435-1445.

Hopgood, B. (2001). History of the Web, *Oxford Brookes University*.

Horng, S. J., Su, M. Y., Yuan, H. C., Tzong, W. K., Rong, J. C., Jui, L. L., & Citra, D. P. (2011). A Novel Intrusion Detection System Based on Hierarchical Clustering and Support Vector Machines. *Expert Systems with Applications*, *38*(1), 306-313.

KDD Cup 1999 Data. (1999). *The UCI KDD Archive* [Data file). Retrieved from http://kdd.ics.uci.edu/databases/KDD Cup99/task.html (last accessed January, 2012).

Hui, J. Y. (2016). Research of Network Intrusion Detection System based on Machine Learning and Rough Set Theory. *Advanced Science and Technology Letters*, 120–124.

Internet Users. (2016). Internet World Stats. Retrieved from https://www.internetworldstats.com/stats.htm (last accessed January, 2017).

Jaisankar, N., Ganapathy, S., & Kannan, A. (2012). Intelligent Intrusion Detection System Using Fuzzy Rough Set Based C4 . 5 Algorithm. *Procceding of the International Conference on Advances in Computing, Communications and Informatics*, India, 596–601.

Janmejay, P., Kamlesh, P., Himanshu, P. (2015). Rough Set Approach for Feature Selection in IDS [Special Issue]. *International Journal of Innovations & Advancement in Computer Science*, 4, 8-11.

Jerzy, W. G. B. (2005). Introduction to Rough Set Theory and Applications. *Real World Applications of Computational Intelligence*, 76-147.

Julisch, K., & Dacier, M. (2002). Mining Intrusion Detection Alarms for Actionable Knowledge. *Procceding of the 8th International Conference on Knowledge Discovery and Data Mining, ACM*, 366–375.

Kabir, E., Hu, J., Wang, H., & Zhuo, G. (2018). A Novel Statistical Technique for Intrusion Detection Systems. *Future Generation Computer Systems, 79*(1).

Kale, S. V., Kale, S. R., Naphade, R. A., & Dande, P. A. A. (2016). A Review of Various Intrusion Detection Approaches. *International Journal of Advanced Research Engineering, Computer Science and Software*, *6*(3), 261–262.

Kaur, H., Singh, G., & Minhas, J. (2013). A Review of Machine Learning Based Anomaly Detection Technique. *International Journal of Computer Applications Technology and Research*, *2*(2), 185-187.

Kaur, K., & Kaur, N. (2015). A Hybrid Approach of Fuzzy C-Mean Clustering and Genetic Algorithm (GA) to Improve Intrusion Detection Rate. *International Journal of Science and Research*, *5*(5), 955-959.

Kausar, N., Samir, B. B., Sulaiman, S. B., Ahmad, I., & Hussain, M. (2012). An Approach Towards Intrusion Detection using PCA Feature Subsets and SVM. *Procceding of the International Conference on Computer & Information Science 2012*, Malaysia, 569-574.

Kavakiotis, I., Tsave, O., Salifoglou, A., Maglaveras, N., Vlahavas, I., & Chouvarda, I. (2017). Machine Learning and Data Mining Methods in Diabetes Research. *Journal of Computational and Structural Biotechnology*, 104–116.

Kerber, R., & Chimerge. (1992). Discretization of Numeric Value. *Procceding of the National Conference of Artificial Intelligent*, California, 123–128. Retrieved from https://sci2s.ugr.es/keel/pdf/algorithm/congreso/1992-Kerber-ChimErge-AAAI92.pdf (last accessed January, 2014).

Koller, D., & Sahami, M. (1996). Toward Optimal Feature Selection. *International Conference on Machine Learning*, Scotland, 284-292.

Kumar, A., Maurya, H. C., & Mishra, R. (2013). A Research Paper on Hybrid Intrusion Detection System. *International Journal and Advanced Technology*, *2*(4), 294-297.

Kumar, M., & Yadav, N. (2015). Fuzzy Rough Sets and Its Application in Data Mining Field. *Journal of Advances in Computer Science and Information Technology*, *2*(3), 237-240.

Kumar, Y., & Dhawan, S. (2012). A Review on Information Flow in Intrusion Detection System. *International Journal of Computational Engineering & Management*, 15(1), 91–96.
Kurgan, L.A., Cios, K. J. (2004). Caim Discretization Algorithm. *IEEE Transactions on Knowledge and Data Engineering,16*(2), 145–153.

Liu, L., Wan, P., Yingmei, W., Songtao, L. (2014). Clustering and Hybrid Genetic Algorithm Based Intrusion Detection Strategy. *Indonesian Journal of Electrical Engineering and Computer Science*, *12*(1), 762-770.

Leandros, A. M., Jianmin, J., Tiago, C. (2014). Integrated OCSVM Mechanism for Intrusion Detection In SCADA Systems. *IET Electronics Letters*, *50*(25), 1935-1936.

Li, J. (2007). Rough Set Based Rule Evaluations and Their Application (Doctoral dissertation, University of Waterloo) Retrieved from https://pdfs.semanticscholar.org/6c1b/9aa86e29577c6bd106c89a3279bc4bb81bd9.pdf (last accessed January, 2013).

Li, J., & Cercone, N. (2005). *Empirical Analysis on the Geriatric Care Dataset Using Rough Sets Theory*. (Report No CS-*2005*-05). Canada: University of Waterloo.

Li, Y., Xia, J., Zhang, S., Yan, J., Xi, X., & Dai, K. (2012). An Efficient Intrusion Detection System Based on Support Vector Machines and Gradually Feature Removal Method. *Expert Systems with Applications*, *39*(1), 424-430.

Lin, S. W., Ying, K. C., Lee, C. Y., & Lee, Z. J. (2012). An Intelligent Algorithm with Feature Selection and Decision Rules Applied to Anomaly Intrusion Detection. *Applied Soft Computing*, *12*(10), 3285-3290.

Liu, H., Hussain, F., Tan, C.L., & Dash, M. (2002). Discretization: An Enabling Technique. *Data Mining and Knowledge Discovery,*6(4), 393–423.

Liyana, N., Shuib, M., Bakar, A. A., & Othman, Z. A. (2011). Performance Study on Data Discretization Techniques Using Nutrition Dataset, *Proceeding of the International Symposium on Computing, Communication, and Control 2009,* Singapore, 304–308.

Majidi, F., Mirzaei, H., Irnapour, T., & Faroughi, F. (2008). A Diversity Creation Method for Ensemble Based Classification: Application in Intrusion Detection Systems, *Proceeding of the7th IEEE International Conference on Cybernetic Intelligent Systems 2008*, UK, 1-5.

Manocha, S., & Irolami, M. A. G. (2007). An Empirical Analysis of the Probabilist C K-Nearest Neighbor Classifier. *Pattern Recognition Letters*, *28*(13), 1818-1824.

Metz, C. (1978). Basic Principles of ROC Analysis. *Seminars in Nuclear Medicine*, *8*(4), 283–298.

Ming, H., Wenying, N., & Xu, L. (2009). An Improved Decision Tree Classification Algorithm Based on ID3 and the Application in Score Analysis. *Procceding of the 21st Annual International Conference On Chinese Control And Decision Conference*, China, 1931–1934.

Mohammed, M. N., & Sulaiman, N. (2012). Intrusion Detection System Based on SVM for WLAN. *Procedia Technology*, 313-317.

Mukherjee, S., & Sharma, N. (2012). Intrusion Detection using Naive Bayes Classifier with Feature Reduction. *Procedia Technology*, 119-128.

Neha, G. R., Dharmaraj, R. P. (2015). Implementation of Network Intrusion Detection System using Variant of Decision Tree Algorithm. *Procceding of the International Conference of Nascent Technologies in the Engineering Field, IEEE*, India, 1-5.

Nguyen, H. S. (1998). Discretization Problem for Rough Sets Methods. Lecture Notes in Computer Science, Vol: 1424. *Procceding of the 1st International Conference on Rough Sets and Current Trend in Computing* (pp. 545-552).

Ohrn, A. (1999). *Discernibility and Rough Sets in Medicine: Tools and Applications* (Doctoral dissertation, Norwegian University of Science and Technology). Retrieved from https://wiki.eecs.yorku.ca/course_archive/2011-12/F/4403/_media/ohrn_thesis.pdf (last accessed January, 2013).

Pang, N. T., & Kumar, V. (2000). *Interestingness Measures for Association Patterns: A Perspective.* (Report No TR00-036). University of Minnesota: KDD 2000 Workshop on Postprocessing in Machine Learning and Data Mining.

Panigrahi, A., & Patra, M. R. (2018). Fuzzy Rough Set Based Network Intrusion Detection with Wrapper Subset Evaluator. *International Journal of Engineering Science Invention (IJESI)*, *7*(2), 51–57.

Pawlak, Z. (1991). Rough Sets, Theoretical Aspects of Reasoning about Data. *Kluwer Academic Publisher.*

Pawlak, Z. (1991). Rough Sets: Theoretical Aspects of Reasoning about Data. *System Theory, Knowledge Engineering and Problem Solving.* Kluwer Academic Publishers. Retrieved from https://www.springer.com/gp/book/9780792314721 (last accessed January, 2014).

Pawlak, Z. (1997). Rough Set Approach to Knowledge-Based Decision Support, *European of Operation Research, 99*(1).

Pawlak, Z., & Skowron, A. (2007). Rough Sets and Boolean Reasoning. *Information Science*, 177*,* 41-73.

Peddabachigari, S., Abraham, A., Gransen, C., & Thomas, J. (2007). Modeling Intrusion Detection System using Hybrid Intelligent Systems. *Network and Computer Applications*, *30*(1), 114-132.

Rahm, E. (2000). Data Cleaning: Problems and Current Approaches. *IEEE Data Engineering Bulletin*, 3-13. Retrieved from https://betterevaluation.org/sites/default/files/data_cleaning.pdf (last accessed March, 2013).

Rampure, V., & Tiwari, A. (2015). A Rough Set Based Feature Selection on KDD CUP 99 Dataset. *Procceding of the Conference of IEEE Region 10 Conference* 2006, China, *8*(1), 149–156.

Repalle, S. A., & Kolluru, V. R. (2017). Intrusion Detection System using AI and Machine Learning Algorithm. *International Research Journal of Engineering and Technology, 4*(12), 1709-1715.

Rizvi, R. S. H., & Keole, R. R. (2015). A Review on Intrusion Detection System. *International Journal of Advance Research in Computer Science and Management Studies*, *3*(3), 22–28.

Sabri, F. N., Norwawi, N. M., & Seman, K. (2011). Identifying False Alarm Rates for Intrusion Detection System with Data Mining. *International Journal of Computer Science and Network Security*, *11*(4), 95-55.

Sadek, R. A., Soliman, M. S., & Elsayed, H. S. (2013). Effective Anomaly Intrusion Detection System based on Neural Network with Indicator Variable and Rough set Reduction. *International Journal of Computer Science Issue*, *10*(6).

Sengupta, N., Sen, J., Sil, J., & Saha, M. (2013). Designing of Online Intrusion Detection System using Rough Set Theory and Q-Learning Algorithm. *Neurocomputing*, 161–168.

Shen, J., Wang, J., & Ai, H. (2012). An Improved Artificial Immune System-Based Network Intrusion Detection by Using Rough Set. *Communications and Network of Scientific Research*, 41–47.

Singh, B. D., Choudhary, N., & Samota, J. (2013). Analysis of Data Mining Classification with Decision Tree Technique. *Global Journal of Computer Science and Technology Software and Data Engineering*. *13*(13), 1-7. Retrieved from https://pdfs.semanticscholar.org/a0e8/f11b9610b45d9e114fcb8c092c9ced2c3f6a.pdf (last accessed January, 2014).

Software, M. (2018). MedCalc: Easy-To-Use Statistical Software. Retrieved from https://www.medcalc.org/manual/roc-curves.php (last accessed September, 2018).

Srikant, R., & Agrawal, R. (1995). Mining Generalized Association Rules. *Procceding of the Conference of 21st Very large Scale Data Bases*, Switzerland, 407-419.

Sujendran, R., & Arunachalam, M. (2015). Hybrid Fuzzy Adaptive Wiener Filtering with Optimization for Intrusion Detection. *Electronic & Telecommunication Research Institute*, *37*(3).

Suresh, C. S., & Anima, N., (2012). Hybridization of Rough Set and Differential Evolution Technique for Optimal Feature Selection. *Procceding of the International Conference on Information Systems Design and Intelligent Applications 2012*, India, 453-460.

Vadim, K. (2018). Overview of Different Approaches to Solving Problems of Data Mining. *Procedia Computer Science*, 234–239.

Wu, H. C., & Huang, S. H. S. (2010). Neural Networks Based Detection of Stepping Stone Intrusion. *Expert Systems with Applications*, *37*(2), 1431-1437.

Wu, S. X., & Banzhaf, W. (2010). The Use of Computational Intelligence in Intrusion Detection Systems: A Review. *Applied Soft Computing*, 1-35.

Wu, S., & Yen, E. (2009). Data Mining Based Intrusion Detectors. *Expert Systems with Applications*, 36(3), 5605-5612.

Yang, Y., Webb, G. I. (2009). Discretization for Naïve Bayes Learning: Managing Discretization Bias and Variance. *Machine Learning*, 39–74.

Ye, C. Z., Yang, J., Geng, D., Zhou, Y., & Chen, N. Y. (2002). Fuzzy Rules to Predict Degree of Malignancy in Brain Glioma. *Medical Biology Computer Engineering*, *40*(2), 145-152.

Yu, B., Byres, E., & Howey, C. (2001). Monitoring Controller's "DNA Sequence" for System Security. *Procceding of the Conference of Emerging Technologies, Instrumentation Systems and Automation Society*, Houston, 1-10.

Zainal, A., Hamid H. Jebur, Mohd, A. M. (2015). Enhancing Rough Set Theory Attributes Selection of Kdd Cup 1999. *Theoretical and Applied Information Technology*, *76*(3), 393-400.

Zhang, X., Jia, L., Shi, H., Tang, Z., & Wang, X. (2012). The Application of Machine Learning Methods to Intrusion Detection. *Proceeding of the Spring Congress on Engineering and Technology 2012*, China, 1-6.

Zimmermann, H. (2001). Fuzzy Set Theory and Its Applications. *Kluwer Academic Publisher*. 4th Edition. Retrieved from https://cours.etsmtl.ca/sys843/REFS/Books/ZimmermannFuzzySetTheory2001.pdf (last accessed January, 2014).

Zweig, M. H., & Campbell, G. (1993). Receiver Operating Characteristic (ROC) Plots a Fundamental Evaluation Tool in Clinical Medicine. *Journal of Clinic Chemical*, 561–577.

APPENDIX A
SAMPLE of KDD CUP 99 DATASET

```
duration,protocol_type,service,flag,src_bytes,dst_bytes,land,wrong_fragment,urgent,hot,num_failed_logins,logged_in,num_compro mised,root_shell,su_attempted,num_root,nu
m_file_creations,num_shells,num_access_files,num_outbound_cmds,1s_host_login,1s_guest_login,count,srv_count,serror_rate,srv_serror_rate,rerror_rate,srv_rerror_rate,same
_srv_rate,diff_srv_rate,srv_diff_host_rate,dst_host_count,dst_host_srv_count,dst_host_same_srv_rate,dst_host_diff_srv_rate,dst_host_same_src_port_rate,dst_host_srv_diff_ho
st_rate,dst_host_serror_rate,dst_host_srv_serror_rate,dst_host_rerror_rate,dst_host_srv_rerror_rate,type_attack
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.01,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,domain_u,SF,29,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,0.00,0.00,0.00,0.00,0.50,1.00,0.00,10,3,0.30,0.30,0.30,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,http,SF,223,185,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,4,4,0.00,0.00,0.00,0.00,1.00,0.00,0.00,71,255,1.00,0.00,0.01,0.01,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,http,SF,230,260,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,19,0.00,0.00,0.00,0.00,1.00,0.00,0.11,3,255,1.00,0.00,0.33,0.07,0.33,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,252,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
1,tcp,smtp,SF,3170,329,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,2,0.00,0.00,0.00,0.00,1.00,0.00,1.00,54,39,0.72,0.11,0.02,0.00,0.02,0.00,0.09,0.13,normal.
0,tcp,http,SF,297,13787,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,177,255,1.00,0.00,0.01,0.01,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,291,3542,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,12,12,0.00,0.00,0.00,0.00,1.00,0.00,0.00,187,255,1.00,0.00,0.01,0.01,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,295,753,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,21,22,0.00,0.00,0.00,0.00,1.00,0.00,0.09,196,255,1.00,0.00,0.01,0.01,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,http,SF,268,9235,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5,5,0.00,0.00,0.00,0.00,1.00,0.00,0.00,58,255,1.00,0.00,0.02,0.05,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,http,SF,223,185,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,3,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,227,8841,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,13,13,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,222,19564,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,22,23,0.00,0.00,0.00,0.00,1.00,0.00,0.09,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,ftp_data,SF,740,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,77,33,0.34,0.08,0.34,0.06,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,ftp_data,SF,35195,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10,10,0.00,0.00,0.00,0.00,1.00,0.00,0.00,92,44,0.43,0.07,0.43,0.05,0.00,0.00,0.00,0.00,normal.
0,tcp,ftp_data,SF,8325,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,20,0.00,0.00,0.00,0.00,1.00,0.00,0.00,103,54,0.49,0.06,0.49,0.04,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,smtp,SF,559,336,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,84,176,0.75,0.08,0.01,0.01,0.01,0.01,0.08,0.05,normal.
0,tcp,http,SF,227,182,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,252,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,317,278,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,3,0.00,0.00,0.00,0.00,1.00,0.00,0.00,192,255,1.00,0.00,0.01,0.04,0.00,0.00,0.00,0.00,normal.
1,tcp,smtp,SF,1661,330,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,3,0.00,0.00,0.00,0.00,1.00,0.00,1.00,172,126,0.37,0.04,0.01,0.02,0.01,0.00,0.02,0.02,normal.
20,tcp,ftp,SF,232,765,0,0,0,4,0,1,0,0,0,0,0,0,0,0,0,1,2,1,0.00,0.00,0.00,0.00,0.50,1.00,0.00,179,48,0.27,0.04,0.01,0.00,0.01,0.02,0.02,0.00,normal.
0,tcp,http,SF,322,680,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,8,0.00,0.00,0.00,0.00,1.00,0.00,0.25,6,255,1.00,0.00,0.17,0.04,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,321,2060,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,18,0.00,0.00,0.00,0.00,1.00,0.00,0.11,8,255,1.00,0.00,0.12,0.03,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,234,14497,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,24,0.00,0.00,0.00,0.00,1.00,0.00,0.12,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,218,261,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,11,32,0.00,0.00,0.00,0.00,1.00,0.00,0.09,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,216,7504,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,20,21,0.00,0.00,0.00,0.00,1.00,0.00,0.10,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,ftp_data,SF,615,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,0.00,0.00,0.00,0.00,0.50,1.00,0.00,174,56,0.32,0.03,0.32,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,312,677,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,9,37,0.00,0.00,0.00,0.00,1.00,0.00,0.11,22,255,1.00,0.00,0.05,0.04,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,307,1528,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,7,7,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,http,SF,246,753,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,9,9,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,http,SF,245,4199,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,smtp,SF,908,330,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,240,101,0.38,0.03,0.00,0.02,0.01,0.00,0.01,0.03,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,252,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,http,SF,232,2367,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,3,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,235,695,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,13,13,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
```

```
0,tcp,http,SF,249,198,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,23,23,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,235,3817,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,11,11,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,243,2211,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,21,21,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,236,3665,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,31,31,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,237,675,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,41,41,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,240,662,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,51,51,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,240,2172,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,59,59,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,315,1528,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,4,4,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,313,5381,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,255,1.00,0.00,1.00,0.01,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00,normal.
1,tcp,smtp,SF,1018,333,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,3,0.00,0.00,0.00,0.00,1.00,0.00,1.00,1,112,1.00,0.00,1.00,0.02,0.00,0.00,0.00,0.00,normal.
1,tcp,smtp,SF,1145,329,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,4,160,0.75,0.50,0.25,0.02,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,http,SF,244,668,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,7,7,0.00,0.00,0.00,0.00,1.00,0.00,0.00,38,255,1.00,0.00,0.03,0.01,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,275,525,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,16,0.00,0.00,0.00,0.00,1.00,0.00,0.19,2,255,1.00,0.00,0.50,0.05,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,247,259,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,22,26,0.00,0.00,0.00,0.00,1.00,0.00,0.12,53,255,1.00,0.00,0.02,0.01,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,245,662,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,31,35,0.00,0.00,0.00,0.00,1.00,0.00,0.09,62,255,1.00,0.00,0.02,0.01,0.00,0.00,0.00,0.00,normal.
0,tcp,ftp_data,SF,884,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,14,39,0.43,0.14,0.43,0.05,0.00,0.00,0.00,0.00,normal.
0,tcp,ftp_data,SF,854,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10,10,0.00,0.00,0.00,0.00,1.00,0.00,0.00,24,49,0.67,0.08,0.67,0.04,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,icmp,eco_i,SF,30,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,3,0.01,0.01,0.01,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,252,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,domain_u,SF,45,110,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,9,0.00,0.00,0.00,0.00,1.00,0.00,0.44,255,246,0.96,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
1,tcp,smtp,SF,1625,377,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,2,0.00,0.00,0.00,0.00,1.00,0.00,1.00,8,160,0.88,0.25,0.12,0.02,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,other,SF,1214,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,1,0.00,0.00,0.00,0.00,0.11,0.56,0.00,25,1,0.04,0.40,0.76,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,299,2178,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,2,238,1.00,0.00,0.50,0.05,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,291,1010,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,12,12,0.00,0.00,0.00,0.00,1.00,0.00,0.00,12,238,1.00,0.00,0.08,0.05,0.00,0.00,0.00,0.00,normal.
0,tcp,ftp_data,SF,252,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,9,38,0.33,0.22,0.33,0.08,0.00,0.00,0.11,0.00,normal.
0,udp,domain_u,SF,43,107,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,5,0.00,0.00,0.00,0.00,1.00,0.00,0.40,255,238,0.93,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.01,0.00,0.00,0.00,0.00,0.00,normal.
28,tcp,ftp,SF,859,2586,0,0,0,18,0,1,0,0,0,0,0,0,0,0,0,1,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,40,4,0.10,0.43,0.03,0.00,0.03,0.00,0.00,0.00,normal.
0,tcp,smtp,SF,764,329,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,12,159,0.83,0.25,0.08,0.01,0.00,0.00,0.00,0.00,normal.
0,tcp,ftp_data,SF,15722,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,19,47,0.63,0.11,0.63,0.06,0.00,0.00,0.05,0.00,normal.
0,tcp,ftp_data,SF,151,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,11,11,0.00,0.00,0.00,0.00,1.00,0.00,0.00,29,54,0.76,0.07,0.76,0.06,0.00,0.00,0.03,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,domain_u,SF,45,45,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,4,0.00,0.00,0.00,0.00,1.00,0.00,0.50,255,236,0.93,0.01,0.01,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,smtp,SF,1145,332,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,14,100,0.79,0.14,0.07,0.03,0.00,0.00,0.00,0.00,normal.
0,udp,domain_u,SF,44,112,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,4,0.00,0.00,0.00,0.00,1.00,0.00,0.50,255,236,0.93,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,smtp,SF,765,325,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,2,0.00,0.00,0.00,0.00,1.00,0.00,1.00,46,91,0.20,0.37,0.02,0.02,0.02,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,domain_u,SF,45,115,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,8,0.00,0.00,0.00,0.00,1.00,0.00,0.25,255,236,0.93,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,smtp,SF,2229,481,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,6,0.02,0.02,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,domain_u,SF,44,44,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,4,0.00,0.00,0.00,0.00,1.00,0.00,0.50,255,236,0.93,0.01,0.01,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,315,938,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,7,0.00,0.00,0.00,0.00,1.00,0.00,0.29,24,255,1.00,0.00,0.04,0.01,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,294,289,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,16,17,0.00,0.00,0.00,0.00,1.00,0.00,0.12,152,255,1.00,0.00,0.01,0.01,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,305,1745,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,23,27,0.00,0.00,0.00,0.00,1.00,0.00,0.11,159,255,1.00,0.00,0.01,0.01,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,http,SF,286,958,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,153,1.00,0.00,1.00,0.03,0.00,0.33,0.00,0.00,normal.
0,tcp,http,SF,278,1731,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,11,11,0.00,0.00,0.00,0.00,1.00,0.00,0.00,11,153,1.00,0.00,0.09,0.03,0.00,0.27,0.00,0.00,normal.
0,tcp,http,SF,315,938,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,6,0.00,0.00,0.00,0.00,1.00,0.00,0.33,43,255,1.00,0.00,0.02,0.01,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,296,921,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,15,16,0.00,0.00,0.00,0.00,1.00,0.00,0.12,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,323,3301,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,26,0.00,0.00,0.00,0.00,1.00,0.00,0.12,50,255,1.00,0.00,0.02,0.01,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,340,247,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,9,0.00,0.00,0.00,0.00,1.00,0.00,0.33,19,255,1.00,0.00,0.05,0.04,0.00,0.01,0.00,0.00,normal.
0,tcp,auth,SF,9,35,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,38,1,0.03,0.11,0.03,0.00,0.00,0.00,0.13,0.00,normal.
```

```
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,tcp,smtp,SF,1483,475,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,2,0.00,0.00,0.00,0.00,1.00,0.00,1.00,255,14,0.05,0.02,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,smtp,SF,910,332,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,27,172,0.96,0.07,0.04,0.01,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,224,2006,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,4,5,0.00,0.00,0.00,0.00,1.00,0.00,0.40,82,255,1.00,0.00,0.01,0.03,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,163,20554,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,12,15,0.00,0.00,0.00,0.00,1.00,0.00,0.13,90,255,1.00,0.00,0.01,0.03,0.00,0.00,0.00,0.00,normal.
0,udp,domain_u,SF,45,115,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,5,0.00,0.00,0.00,0.00,1.00,0.00,0.40,255,247,0.97,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,299,1401,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,4,6,0.00,0.00,0.00,0.00,1.00,0.00,0.50,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,296,283,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,11,13,0.00,0.00,0.00,0.00,1.00,0.00,0.15,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,296,188,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,23,0.00,0.00,0.00,0.00,1.00,0.00,0.13,14,255,1.00,0.00,0.07,0.05,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,289,1743,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,27,32,0.00,0.00,0.00,0.00,1.00,0.00,0.09,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,domain_u,SF,46,117,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,4,0.00,0.00,0.00,0.00,0.67,0.67,0.50,255,249,0.98,0.01,0.01,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,domain_u,SF,46,117,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,8,0.00,0.00,0.00,0.00,1.00,0.00,0.25,255,251,0.98,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,ftp_data,SF,192,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,69,71,0.33,0.26,0.52,0.03,0.01,0.01,0.00,0.00,normal.
0,udp,domain_u,SF,45,115,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,0.00,0.00,0.00,0.00,1.00,0.00,1.00,255,251,0.98,0.01,0.01,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,domain_u,SF,45,114,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,0.00,0.00,0.00,0.00,1.00,0.00,1.00,255,251,0.98,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,335,606,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,10,10,0.00,0.00,0.00,0.00,1.00,0.00,0.00,126,255,1.00,0.00,0.01,0.05,0.01,0.00,0.00,0.00,normal.
0,tcp,http,SF,342,396,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,20,20,0.00,0.00,0.00,0.00,1.00,0.00,0.00,136,255,1.00,0.00,0.01,0.05,0.01,0.00,0.00,0.00,normal.
0,tcp,http,SF,354,2650,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,29,30,0.00,0.00,0.00,0.00,1.00,0.00,0.07,145,255,1.00,0.00,0.01,0.05,0.01,0.00,0.00,0.00,normal.
0,tcp,http,SF,274,349,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,93,255,1.00,0.00,0.01,0.03,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,257,9885,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,12,12,0.00,0.00,0.00,0.00,1.00,0.00,0.00,103,255,1.00,0.00,0.01,0.03,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,270,345,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,22,22,0.00,0.00,0.00,0.00,1.00,0.00,0.00,113,255,1.00,0.00,0.01,0.03,0.00,0.00,0.00,0.00,normal.
0,icmp,ecr_i,SF,30,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,75,75,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,252,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,ftp_data,SF,190,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,43,13,0.30,0.09,0.30,0.00,0.00,0.00,0.05,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,252,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,203,294,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,7,9,0.00,0.00,0.00,0.00,1.00,0.00,0.22,54,255,1.00,0.00,0.02,0.05,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,211,631,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,15,19,0.00,0.00,0.00,0.00,1.00,0.00,0.11,62,255,1.00,0.00,0.02,0.05,0.00,0.00,0.00,0.00,normal.
0,udp,domain_u,SF,45,110,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,0.00,0.00,0.00,0.00,1.00,0.00,1.00,255,251,0.98,0.01,0.01,0.00,0.00,0.00,0.00,0.00,normal.
6153,tcp,IRC,RSTR,852,6787,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,1.00,1.00,1.00,0.00,0.00,12,4,0.33,0.17,0.08,0.00,0.00,0.00,0.33,1.00,normal.
0,tcp,http,SF,373,412,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,4,4,0.00,0.00,0.00,0.00,1.00,0.00,0.00,149,255,1.00,0.00,0.01,0.05,0.01,0.00,0.00,0.00,normal.
0,udp,domain_u,SF,44,115,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,3,0.00,0.00,0.00,0.00,1.00,0.00,0.67,255,251,0.98,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,252,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack.
0,udp,private,SF,105,145,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,252,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,203,294,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,7,10,0.00,0.00,0.00,0.00,1.00,0.00,0.20,70,255,1.00,0.00,0.01,0.05,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,204,2199,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,14,18,0.00,0.00,0.00,0.00,1.00,0.00,0.11,77,255,1.00,0.00,0.01,0.05,0.00,0.00,0.00,0.00,normal.
0,tcp,X11,S1,286040,383476,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1.00,1.00,0.00,0.00,1.00,0.00,0.00,53,1,0.02,0.11,0.02,0.00,0.02,1.00,0.09,0.00,normal.
1,tcp,smtp,SF,1676,333,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,31,157,0.84,0.10,0.03,0.01,0.00,0.00,0.00,0.00,normal.
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,258,1432,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5,5,0.00,0.00,0.00,0.00,1.00,0.00,0.00,5,255,1.00,0.00,0.20,0.03,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,316,5141,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,13,14,0.00,0.00,0.00,0.00,1.00,0.00,0.14,13,255,1.00,0.00,0.08,0.03,0.00,0.00,0.00,0.00,normal.
```

APPENDIX B

SAMPLE of 30 of 25590 EFB DISRETIZATION DATA of DECISION TABLE

```
[*, 1)  tcp      http     SF       [239, 355)         [1696, *)          0        0        [*, 1)   [*,   1)
        [*, 1)   1        [*, 1)   0        0         [*, 1)   [*, 1)    [*, 1)   [*, 1)   0        0
        0        [4, 17)  [5, 21)  0        0         0        0         1        0        0        [*,  80)
        [255, *) 1        0        [0.01, 0.09)       [0.03, *) 0        0        0        0
        normal.
[*, 1)  tcp      http     SF       [239, 355)         [1696, *)          0        0        [*, 1)   [*,   1)
        [*, 1)   1        [*, 1)   0        0         [*, 1)   [*, 1)    [*, 1)   [*, 1)   0        0
        0        [*, 4)   [*, 5)   0        0         0        0         1        0        0        [*,  80)
        [255, *) 1        0        [0.01, 0.09)       [0.03, *) 0        0        0        0
        normal.
[*, 1)  tcp      http     SF       [239, 355)         [1696, *)          0        0        [*, 1)   [*,   1)
        [*, 1)   1        [*, 1)   0        0         [*, 1)   [*, 1)    [*, 1)   [*, 1)   0        0
        0        [4, 17)  [5, 21)  0        0         0        0         1        0        0        [*,  80)
        [255, *) 1        0        [0.01, 0.09)       [0.01, 0.03)       0        0        0        0
        normal.
[*, 1)  tcp      http     SF       [239, 355)         [281, 1696)        0        0        [*, 1)   [*,   1)
        [*, 1)   1        [*, 1)   0        0         [*, 1)   [*, 1)    [*, 1)   [*, 1)   0        0
        0        [4, 17)  [5, 21)  0        0         0        0         1        0        0        [*,  80)
        [255, *) 1        0        [0.01, 0.09)       [0.01, 0.03)       0        0        0        0
        normal.
[*, 1)  tcp      http     SF       [239, 355)         [1696, *)          0        0        [*, 1)   [*,   1)
        [*, 1)   1        [*, 1)   0        0         [*, 1)   [*, 1)    [*, 1)   [*, 1)   0        0
        0        [4, 17)  [5, 21)  0        0         0        0         1        0        0        [*,  80)
        [255, *) 1        0        [0.09, *) [0.03, *) 0       0         0        0        normal.
[*, 1)  tcp      http     SF       [239, 355)         [1696, *)          0        0        [*, 1)   [*,   1)
        [*, 1)   1        [*, 1)   0        0         [*, 1)   [*, 1)    [*, 1)   [*, 1)   0        0
        0        [*, 4)   [*, 5)   0        0         0        0         1        0        0        [*,  80)
        [255, *) 1        0        [0.09, *) [0.03, *) 0       0         0        0        normal.
[*, 1)  tcp      http     SF       [*, 239)  [1696, *)         0         0        [*, 1)   [*, 1)   [*,   1)
        1        [*, 1)   0        0        [*, 1)    [*, 1)   [*, 1)    [*, 1)   0        0        0
        [*, 4)   [*, 5)   0        0        0         0        1         0        0        [*, 80)  [255,
*)      1        0        [0.09, *) [0.03, *) 0       0        0         0        normal.
[*, 1)  tcp      http     SF       [239, 355)         [1696, *)          0        0        [*, 1)   [*,   1)
        [*, 1)   1        [*, 1)   0        0         [*, 1)   [*, 1)    [*, 1)   [*, 1)   0        0
        0        [4, 17)  [5, 21)  0        0         0        0         1        0        0        [*,  80)
        [255, *) 1        0        [0.01, 0.09)       [0.03, *) 0        0        0        0
        normal.
```

```
[*, 1)    tcp       http      SF        [239, 355)          [1696, *)           0         0         [*, 1)    [*,   1)
          [*, 1)    1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0
          0         [4, 17)   [5, 21)   0         0         0         0         1         0         0         [*,  80)
          [255, *)  1         0         [0.01, 0.09)        [0.03, *) 0         0         0         0
          normal.
[*, 1)    tcp       http      SF        [239, 355)          [1696, *)           0         0         [*, 1)    [*,   1)
          [*, 1)    1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0
          0         [4, 17)   [5, 21)   0         0         0         0         1         0         0         [*,  80)
          [255, *)  1         0         [0.01, 0.09)        [0.03, *) 0         0         0         0
          normal.
[*, 1)    tcp       http      SF        [*, 239)  [281, 1696)         0         0         [*, 1)    [*, 1)    [*,   1)
          1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0         0
          [4, 17)   [5, 21)   0         0         0         0         1         0         0         [*, 80)   [255,
*)        1         0         [0.09, *) [0.03, *) 0         0         0         0         normal.
[*, 1)    tcp       http      SF        [*, 239)  [281, 1696)         0         0         [*, 1)    [*, 1)    [*,   1)
          1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0         0
          [4, 17)   [*, 5)    0         0         0         0         1         0         0         [*, 80)   [255,
*)        1         0         [0.09, *) [0.03, *) 0         0         0         0         normal.
[*, 1)    tcp       http      SF        [*, 239)  [281, 1696)         0         0         [*, 1)    [*, 1)    [*,   1)
          1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0         0
          [4, 17)   [5, 21)   0         0         0         0         1         0         0         [*, 80)   [255,
*)        1         0         [0.01, 0.09)        [0.03, *) 0         0         0         0         normal.
[*, 1)    tcp       http      SF        [*, 239)  [281, 1696)         0         0         [*, 1)    [*, 1)    [*,   1)
          1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0         0
          [4, 17)   [5, 21)   0         0         0         0         1         0         0         [*, 80)   [255,
*)        1         0         [0.01, 0.09)        [0.03, *) 0         0         0         0         normal.
[*, 1)    tcp       http      SF        [239, 355)          [1696, *)           0         0         [*, 1)    [*,   1)
          [*, 1)    1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0
          0         [*, 4)    [*, 5)    0         0         0         0         1         0         0         [*,  80)
          [255, *)  1         0         [0.01, 0.09)        [0.03, *) 0         0         0         0
          normal.
[*, 1)    tcp       http      SF        [*, 239)  [1696, *)           0         0         [*, 1)    [*, 1)    [*,   1)
          1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0         0
          [4, 17)   [5, 21)   0         0         0         0         1         0         0         [*, 80)   [255,
*)        1         0         [0.01, 0.09)        [0.03, *) 0         0         0         0         normal.
[*, 1)    tcp       http      SF        [239, 355)          [1696, *)           0         0         [*, 1)    [*,   1)
          [*, 1)    1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0
          0         [*, 4)    [*, 5)    0         0         0         0         1         0         0         [80,
255)      [255, *)  1         0         [0.01, 0.09)        [0.03, *) 0         0         0         0
          normal.
```

[*, 1)	tcp	http	SF	[*, 239)	[1696, *)		0	0	[*, 1)	[*, 1)	[*, 1)
	1	[*, 1)	0	0	[*, 1)	[*, 1)	[*, 1)	[*, 1)	0	0	0
	[4, 17)	[5, 21)	0	0	0	0	1	0	0	[*, 80)	[255,
*)	1	0	[0.09, *)	[0.03, *)	0	0	0	0	normal.		
[*, 1)	tcp	http	SF	[239, 355)		[281, 1696)		0	0	[*, 1)	[*, 1)
	[*, 1)	1	[*, 1)	0	0	[*, 1)	[*, 1)	[*, 1)	[*, 1)	0	0
	0	[4, 17)	[5, 21)	0	0	0	0	1	0	0	[*, 80)
	[255, *)	1	0	[0.01, 0.09)		[0.03, *)	0	0	0	0	
	normal.										
[*, 1)	tcp	http	SF	[239, 355)		[281, 1696)		0	0	[*, 1)	[*, 1)
	[*, 1)	1	[*, 1)	0	0	[*, 1)	[*, 1)	[*, 1)	[*, 1)	0	0
	0	[17, *)	[21, *)	0	0	0	0	1	0	0	[*, 80)
	[255, *)	1	0	[0.01, 0.09)		[0.03, *)	0	0	0	0	
	normal.										
[*, 1)	tcp	http	SF	[239, 355)		[281, 1696)		0	0	[*, 1)	[*, 1)
	[*, 1)	1	[*, 1)	0	0	[*, 1)	[*, 1)	[*, 1)	[*, 1)	0	0
	0	[4, 17)	[5, 21)	0	0	0	0	1	0	0	[*, 80)
	[255, *)	1	0	[0.01, 0.09)		[0.03, *)	0	0	0	0	
	normal.										
[*, 1)	tcp	http	SF	[239, 355)		[281, 1696)		0	0	[*, 1)	[*, 1)
	[*, 1)	1	[*, 1)	0	0	[*, 1)	[*, 1)	[*, 1)	[*, 1)	0	0
	0	[4, 17)	[5, 21)	0	0	0	0	1	0	0	[*, 80)
	[255, *)	1	0	[0.01, 0.09)		[0.03, *)	0	0	0	0	
	normal.										
[*, 1)	tcp	http	SF	[*, 239)	[*, 281)	0	0	[*, 1)	[*, 1)	[*, 1)	1
	[*, 1)	0	0	[*, 1)	[*, 1)	[*, 1)	[*, 1)	0	0	0	[17, *)
	[21, *)	0	0	0	0	1	0	0	[*, 80)	[255, *)	1
	0	[0.01, 0.09)		[0.03, *)	0	0	0	0	normal.		
[*, 1)	tcp	http	SF	[*, 239)	[1696, *)		0	0	[*, 1)	[*, 1)	[*, 1)
	1	[*, 1)	0	0	[*, 1)	[*, 1)	[*, 1)	[*, 1)	0	0	0
	[*, 4)	[*, 5)	0	0	0	0	1	0	0	[*, 80)	[255,
*)	1	0	[0.09, *)	[0.03, *)	0	0	0	0	normal.		
[*, 1)	tcp	http	SF	[239, 355)		[281, 1696)		0	0	[*, 1)	[*, 1)
	[*, 1)	1	[*, 1)	0	0	[*, 1)	[*, 1)	[*, 1)	[*, 1)	0	0
	0	[4, 17)	[5, 21)	0	0	0	0	1	0	0	[*, 80)
	[255, *)	1	0	[0.09, *)	[0.03, *)	0	0	0	0	normal.	

```
[*, 1)    tcp       http      SF        [239, 355)          [281, 1696)         0         0         [*, 1)    [*,    1)
          [*, 1)    1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0
          0         [17, *)   [5, 21)   0         0         0         0         1         0         0         [*,  80)
          [255, *)  1         0         [0.01, 0.09)        [0.03, *) 0         0         0         0
          normal.
[*, 1)    tcp       http      SF        [239, 355)          [281, 1696)         0         0         [*, 1)    [*,    1)
          [*, 1)    1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0
          0         [17, *)   [21, *)   0         0         0         0         1         0         0         [*,  80)
          [255, *)  1         0         [0.01, 0.09)        [0.03, *) 0         0         0         0
          normal.
[*, 1)    tcp       http      SF        [239, 355)          [281, 1696)         0         0         [*, 1)    [*,    1)
          [*, 1)    1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0
          0         [4, 17)   [5, 21)   0         0         0         0         1         0         0         [*,  80)
          [255, *)  1         0         [0.01, 0.09)        [0.03, *) 0         0         0         0
          normal.
[*, 1)    tcp       http      SF        [*, 239)  [281, 1696)         0         0         [*, 1)    [*, 1)    [*,    1)
          1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0         0
          [*, 4)    [*, 5)    0         0         0         0         1         0         0         [*, 80)   [255,
*)        1         0         [0.01, 0.09)        [0.03, *) 0         0         0         0         normal.
[*, 1)    tcp       http      SF        [*, 239)  [281, 1696)         0         0         [*, 1)    [*, 1)    [*,    1)
          1         [*, 1)    0         0         [*, 1)    [*, 1)    [*, 1)    [*, 1)    0         0         0
          [4, 17)   [5, 21)   0         0         0         0         1         0         0         [*, 80)   [255,
*)        1         0         [0.01, 0.09)        [0.01, 0.03)        0         0         0         0
          normal.
```

APPENDIX C
REDUCT from DECISION TABLE

{service, flag, src_bytes, dst_bytes, hot, count, srv_count, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate} 100 13

{service, src_bytes, dst_bytes, hot, count, srv_count, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_rerror_rate} 100 13

{service, src_bytes, dst_bytes, hot, count, srv_count, rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate} 100 14

{duration, service, src_bytes, dst_bytes, hot, count, srv_count, srv_rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate} 100 14

{service, src_bytes, dst_bytes, hot, logged_in, count, srv_count, srv_serror_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate} 100 14

{service, src_bytes, dst_bytes, hot, count, srv_count, srv_serror_rate, srv_rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate} 100 14

{service, src_bytes, dst_bytes, hot, count, srv_count, srv_serror_rate, rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate} 100 14

{service, src_bytes, dst_bytes, hot, count, srv_count, srv_rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate} 100 14

{service, src_bytes, dst_bytes, hot, count, srv_count, serror_rate, srv_rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate} 100 14

{service, src_bytes, dst_bytes, hot, count, srv_count, serror_rate, rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate} 100 14

{service, src_bytes, dst_bytes, hot, count, srv_count, rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_srv_serror_rate} 100 14

{duration, service, src_bytes, dst_bytes, hot, count, srv_count, rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate} 100 14

{service, src_bytes, dst_bytes, hot, logged_in, count, srv_count, serror_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate} 100 14

{service, src_bytes, dst_bytes, hot, logged_in, count, srv_count, srv_rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_srv_serror_rate} 100 15

{duration, service, src_bytes, dst_bytes, hot, logged_in, count, srv_count, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate} 100 15

APPENDIX D
RULES in DECISION TABLE

service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_rerror_rate(0) => type_attack(smurf.) 4959 4959 1.0 0.206659 0.997586 1.0 14 1

service(ecr_i) AND flag(SF) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(smurf.) 4959 4959 1.0 0.206659 0.997586 1.0 14 1

duration([*, 1)) AND service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND srv_rerror_rate(0) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(smurf.) 4959 4959 1.0 0.206659 0.997586 1.0 15 1

service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND rerror_rate(0) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_srv_serror_rate(0) => type_attack(smurf.) 4959 4959 1.0 0.206659 0.997586 1.0 15 1

service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND srv_serror_rate(0) AND srv_rerror_rate(0) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(smurf.) 4959 4959 1.0 0.206659 0.997586 1.0 15 1

service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND srv_rerror_rate(0) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_serror_rate(0) => type_attack(smurf.) 4959 4959 1.0 0.206659 0.997586 1.0 15 1 service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND srv_serror_rate(0) AND rerror_rate(0) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(smurf.) 4959 4959 1.0 0.206659 0.997586 1.0 15 1

service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND rerror_rate(0) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_serror_rate(0) => type_attack(smurf.) 4959 4959 1.0 0.206659 0.997586 1.0 15 1

service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND srv_rerror_rate(0) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_srv_serror_rate(0) => type_attack(smurf.) 4959 4959 1.0 0.206659 0.997586 1.0 15 1

service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND serror_rate(0) AND srv_rerror_rate(0) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(smurf.) 4959 4959 1.0 0.206659 0.997586 1.0 15 1

service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND serror_rate(0) AND rerror_rate(0) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(smurf.) 4959 4959 1.0 0.206659 0.997586 1.0 15 1

service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND logged_in(0) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND srv_serror_rate(0) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_srv_serror_rate(0) => type_attack(smurf.) 4959 4959 1.0 0.206659 0.997586 1.0 16 1 duration([*, 1)) AND service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND logged_in(0) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_srv_rerror_rate(0) => type_attack(smurf.) 4959

```
        4959    1.0     0.206659        0.997586        1.0     16      1
```

duration([*, 1)) AND service(ecr_i) AND src_bytes([358, *)) AND dst_bytes([*, 279)) AND hot([*, 1)) AND num_root([*, 1)) AND count([17, *)) AND srv_count([23, *)) AND rerror_rate(0) AND srv_diff_host_rate(0) AND dst_host_count([255, *)) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_srv_rerror_rate(0) => type_attack(smurf.) 4959

```
        4959    1.0     0.206659        0.997586        1.0     16      1
```

APPENDIX E

REDUCT of SIMPLIFICATION DECISION TABLE

{duration, src_bytes}	100	2
{protocol_type, dst_host_count, dst_host_srv_count}	100	3
{duration, dst_bytes}	100	2
{dst_host_srv_count, dst_host_same_src_port_rate}	100	2
{protocol_type, src_bytes, srv_count}	100	3
{protocol_type, srv_count}	100	2
{service, dst_host_srv_diff_host_rate}	100	2
{service, src_bytes}	100	2
{service, src_bytes, count}	100	3
{src_bytes, hot, dst_host_same_src_port_rate}	100	3
{count, dst_host_count}	100	2
{protocol_type, src_bytes, hot, dst_host_srv_count}	100	4
{service, flag}	100	2
{duration, service}	100	2
{service, count}	100	2
{service, dst_host_count}	100	2
{service, src_bytes, dst_host_count}	100	3
{duration, service, dst_host_count}	100	3
{duration, num_root}	100	2
{protocol_type, count}	100	2
{protocol_type, same_srv_rate}	100	2
{duration, protocol_type}	100	2
{service, dst_host_same_srv_rate}	100	2
{src_bytes, num_file_creations}	100	2
{service, dst_host_diff_srv_rate}	100	2
{dst_host_count, dst_host_srv_diff_host_rate}	100	2
{flag, dst_host_same_srv_rate}	100	2
{service, hot}	100	2
{service, src_bytes, dst_host_rerror_rate}	100	3
{duration, dst_host_srv_rerror_rate}	100	2
{service, dst_bytes, dst_host_srv_diff_host_rate}	100	3
{srv_count, dst_host_same_src_port_rate}	100	2
{duration, hot, num_access_files}	100	3
{dst_host_count, dst_host_srv_count}	100	2
{service, srv_diff_host_rate, dst_host_srv_count, dst_host_diff_srv_rate}	100	4
{service, src_bytes, dst_host_same_src_port_rate}	100	3
{service, src_bytes, dst_bytes, hot}	100	4

{service, dst_bytes, dst_host_same_srv_rate}	100	3
{duration, dst_bytes, dst_host_srv_diff_host_rate}	100	3
{protocol_type, srv_diff_host_rate, dst_host_srv_diff_host_rate}	100	3
{service, diff_srv_rate}	100	2
{service, num_file_creations}	100	2
{service, srv_count}	100	2
{service, src_bytes, num_root}	100	3
{service, num_access_files}	100	2
{duration, service, flag}	100	3
{dst_bytes, dst_host_same_src_port_rate}	100	2
{protocol_type, dst_bytes, dst_host_srv_count}	100	3
{dst_bytes, dst_host_srv_count}	100	2
{src_bytes, hot, count, dst_host_srv_count}	100	4
{protocol_type, count, srv_count}	100	3
{duration, srv_count}	100	2
{service, dst_bytes}	100	2
{hot, srv_count, dst_host_same_src_port_rate}	100	3
{dst_bytes, srv_count, dst_host_same_src_port_rate}	100	3
{src_bytes, dst_bytes, hot, count, srv_count, dst_host_same_src_port_rate}	100	6
{flag, src_bytes}	100	2
{service, hot, count}	100	3
{dst_host_count, dst_host_same_src_port_rate}	100	2
{service, src_bytes, hot}	100	3
{duration, dst_host_count}	100	2
{src_bytes, dst_bytes}	100	2
{service, dst_host_same_src_port_rate}	100	2
{service, flag, dst_host_srv_diff_host_rate}	100	3
{duration, service, count}	100	3
{src_bytes, dst_bytes, hot, count, srv_count, dst_host_count}	100	6
{hot, count, srv_count}	100	3
{dst_bytes, hot}	100	2
{duration, diff_srv_rate}	100	2
{num_access_files, srv_count}	100	2
{protocol_type, src_bytes, hot, srv_diff_host_rate}	100	4
{src_bytes, count, dst_host_count}	100	3
{num_access_files, count}	100	2
{duration, num_file_creations}	100	2
{count, dst_host_srv_diff_host_rate}	100	2
{protocol_type, hot, count, dst_host_srv_count}	100	4
{dst_bytes, srv_count, dst_host_count}	100	3

{duration, root_shell, num_access_files}	100	3
{service, flag, dst_host_same_srv_rate}	100	3
{flag, srv_diff_host_rate}	100	2
{srv_count, dst_host_srv_rerror_rate}	100	2
{flag, dst_host_srv_count}	100	2
{flag, dst_host_count}	100	2
{duration, service, dst_host_same_src_port_rate}	100	3
{service, src_bytes, srv_diff_host_rate}	100	3
{duration, num_compromised}	100	2
{num_compromised, srv_diff_host_rate}	100	2
{duration, src_bytes, hot}	100	3
{src_bytes, num_root, srv_diff_host_rate, dst_host_count}	100	4
{service, src_bytes, dst_host_count, dst_host_same_srv_rate, dst_host_srv_diff_host_rate}	100	5
{flag, srv_count}	100	2
{flag, srv_rerror_rate}	100	2
{flag, count}	100	2
{srv_count, dst_host_rerror_rate}	100	2
{dst_host_srv_count, dst_host_rerror_rate}	100	2
{num_compromised, num_file_creations}	100	2
{duration, service, same_srv_rate}	100	3
{service, num_root}	100	2
{num_root, dst_host_count}	100	2
{num_access_files, dst_host_count}	100	2
{src_bytes, srv_diff_host_rate, dst_host_count}	100	3
{root_shell, num_access_files}	100	2
{service, dst_host_srv_serror_rate}	100	2
{service, srv_diff_host_rate}	100	2
{logged_in, dst_host_srv_diff_host_rate}	100	2
{hot, 1s_guest_login}	100	2
{service, serror_rate}	100	2
{service, hot, num_compromised}	100	3
{src_bytes, srv_diff_host_rate, dst_host_same_src_port_rate}	100	3
{duration, dst_host_srv_diff_host_rate}	100	2
{service, src_bytes, dst_host_srv_diff_host_rate}	100	3
{duration, src_bytes, dst_host_srv_diff_host_rate}	100	3
{service, dst_bytes, count}	100	3
{duration, service, src_bytes}	100	3
{duration, dst_host_srv_count}	100	2
{hot, srv_count, dst_host_count}	100	3

APPENDIX F
SIGNIFICANCE RULES of SIMPLIFICATION DECISION TABLE

service(ecr_i) AND src_bytes([358, *)) => type_attack(smurf.) 4971 4971 1.0 0.20716 1.0 1.0 2 1

service(telnet) AND src_bytes([240, 358)) AND dst_host_same_src_port_rate([0.09, *)) => type_attack(loadmodule.) 1 1 1.0 0.000042 1.0 1.0 3 1

src_bytes([*, 240)) AND hot([1, 3)) AND count([*, 4)) AND dst_host_same_src_port_rate([*, 0.01)) => type_attack(phf.) 1 1 1.0 0.000042 1.0 1.0 4 1

dst_bytes([279, 1695)) AND hot([1, 3)) AND count([*, 4)) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(loadmodule.) 1 1 1.0 0.000042 1.0 1.0 4 1

service(ecr_i) AND srv_count([23, *)) => type_attack(smurf.) 4970 4970 1.0 0.207118 0.999799 1.0 2 1

service(telnet) AND dst_bytes([279, 1695)) AND dst_host_same_srv_rate(1) => type_attack(loadmodule.) 1 1 1.0 0.000042 1.0 1.0 3 1

src_bytes([240, 358)) AND hot([1, 3)) AND dst_host_same_src_port_rate([0.09, *)) => type_attack(loadmodule.) 1 1 1.0 0.000042 1.0 1.0 3 1

src_bytes([*, 240)) AND dst_bytes([1695, *)) AND hot([1, 3)) AND dst_host_count([255, *)) => type_attack(phf.) 1 1 1.0 0.000042 1.0 1.0 4 1

dst_bytes([*, 279)) AND count([17, *)) AND srv_count([*, 6)) => type_attack(portsweep.) 18 18 1.0 0.00075 1.0 1.0 3 1

service(telnet) AND dst_bytes([279, 1695)) AND dst_host_same_src_port_rate([0.09, *)) => type_attack(loadmodule.) 1 1 1.0 0.000042 1.0 1.0 3 1

service(finger) AND flag(S0) => type_attack(land.) 1 1 1.0 0.000042 1.0 1.0 2 1

service(ecr_i) AND dst_host_count([255, *)) => type_attack(smurf.) 4971 4971 1.0 0.20716 1.0 1.0 2 1

service(ecr_i) AND count([17, *)) => type_attack(smurf.) 4970 4970 1.0 0.207118 0.999799 1.0 2 1

service(http) AND src_bytes([358, *)) AND hot([1, 3)) => type_attack(back.) 792 792 1.0 0.033006 0.997481 1.0 3 1

service(telnet) AND src_bytes([240, 358)) AND hot([1, 3)) => type_attack(loadmodule.) 1 1 1.0 0.000042 1.0 1.0 3 1

service(telnet) AND dst_bytes([279, 1695)) AND hot([1, 3)) => type_attack(loadmodule.) 1 1 1.0 0.000042 1.0 1.0 3 1

service(telnet) AND src_bytes([240, 358)) AND dst_host_same_srv_rate(1) => type_attack(loadmodule.) 1 1 1.0 0.000042 1.0 1.0 3 1

num_root([5, *)) AND dst_host_same_src_port_rate([*, 0.01)) => type_attack(imap.) 1 1 1.0 0.000042 1.0 1.0 2 1

dst_bytes([1695, *)) AND num_root([5, *)) => type_attack(imap.) 1 1 1.0 0.000042 1.0 1.0 2 1

count([4, 17)) AND dst_host_count([81, 255)) AND dst_host_same_srv_rate(0) AND dst_host_same_src_port_rate([0.01, 0.09)) => type_attack(satan.) 2 2 1.0 0.000083 1.0 1.0 4 1

service(telnet) AND src_bytes([240, 358)) AND dst_bytes([279, 1695)) => type_attack(loadmodule.) 1 1 1.0 0.000042 1.0 1.0 3 1

service(imap4) => type_attack(imap.) 1 1 1.0 0.000042 1.0 1.0 1 1

```
service(finger) AND dst_host_same_srv_rate(1) AND dst_host_diff_srv_rate(0) => type_attack(land.)     1          1
          1.0        0.000042  1.0        1.0        3          1
flag(S0) AND srv_diff_host_rate(0) AND dst_host_same_srv_rate(1) => type_attack(land.)     1          1          1.0
          0.000042  1.0        1.0        3          1
src_bytes([*, 240)) AND hot([1, 3)) AND count([*, 4)) AND dst_host_count([255, *)) => type_attack(phf.)          1
          1          1.0        0.000042  1.0        1.0        4          1
count([4, 17)) AND srv_count([*, 6)) AND dst_host_same_srv_rate(0) AND dst_host_same_src_port_rate([0.01, 0.09)) =>
type_attack(satan.)  2          2          1.0        0.000083  1.0        1.0        4          1
service(telnet) AND hot([1, 3)) AND dst_host_count([*, 81)) AND dst_host_srv_diff_host_rate([0.03, *)) =>
type_attack(loadmodule.)       1          1          1.0        0.000042  1.0        1.0        4          1
service(telnet) AND hot([1, 3)) AND dst_host_diff_srv_rate(0) AND dst_host_srv_diff_host_rate([0.03, *)) =>
type_attack(loadmodule.)       1          1          1.0        0.000042  1.0        1.0        4          1
service(telnet) AND hot([1, 3)) AND dst_host_same_srv_rate(1) AND dst_host_srv_diff_host_rate([0.03, *)) =>
type_attack(loadmodule.)       1          1          1.0        0.000042  1.0        1.0        4          1
service(telnet) AND hot([1, 3)) AND num_root([*, 1)) AND dst_host_count([*, 81)) => type_attack(loadmodule.)   1
          1          1.0        0.000042  1.0        1.0        4          1
src_bytes([*, 240)) AND dst_bytes([1695, *)) AND hot([1, 3)) AND dst_host_same_src_port_rate([*, 0.01)) =>
type_attack(phf.)    1          1          1.0        0.000042  1.0        1.0        4          1
src_bytes([240, 358)) AND dst_bytes([279, 1695)) AND hot([1, 3)) AND count([*, 4)) => type_attack(loadmodule.)
          1          1          1.0        0.000042  1.0        1.0        4          1
flag(S0) AND srv_diff_host_rate(0) AND dst_host_count([*, 81)) => type_attack(land.)        1          1          1.0
          0.000042  1.0        1.0        3          1
flag(S0) AND srv_diff_host_rate(0) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(land.)     1          1
          1.0        0.000042  1.0        1.0        3          1
flag(S0) AND count([*, 4)) AND srv_diff_host_rate(0) => type_attack(land.)        1          1          1.0
          0.000042  1.0        1.0        3          1
src_bytes([*, 240)) AND hot([1, 3)) AND srv_diff_host_rate(1) => type_attack(phf.)          1          1          1.0
          0.000042  1.0        1.0        3          1
src_bytes([358, *)) AND hot([1, 3)) AND num_root([*, 1)) AND dst_host_same_srv_rate(1) => type_attack(back.) 792
          792        1.0        0.033006  0.997481  1.0        4          1
src_bytes([358, *)) AND dst_bytes([1695, *)) AND hot([1, 3)) AND num_root([*, 1)) => type_attack(back.)       792
          792        1.0        0.033006  0.997481  1.0        4          1
service(telnet) AND src_bytes([240, 358)) AND num_root([*, 1)) AND dst_host_count([*, 81)) AND
dst_host_srv_diff_host_rate([0.03, *)) => type_attack(loadmodule.)        1          1          1.0        0.000042  1.0
          1.0        5          1
src_bytes([358, *)) AND hot([1, 3)) AND num_root([*, 1)) AND dst_host_same_srv_rate(1) AND
dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(back.)    792        792        1.0        0.033006  0.997481  1.0
          5          1
service(telnet) AND src_bytes([240, 358)) AND count([*, 4)) AND dst_host_count([*, 81)) AND
dst_host_srv_diff_host_rate([0.03, *)) => type_attack(loadmodule.)        1          1          1.0        0.000042  1.0
          1.0        5          1
src_bytes([358, *)) AND dst_bytes([1695, *)) AND hot([1, 3)) AND num_root([*, 1)) AND dst_host_diff_srv_rate(0) =>
type_attack(back.)   792        792        1.0        0.033006  0.997481  1.0        5          1
src_bytes([358, *)) AND dst_bytes([1695, *)) AND hot([1, 3)) AND num_root([*, 1)) AND dst_host_srv_diff_host_rate([*,
0.01)) => type_attack(back.)   792        792        1.0        0.033006  0.997481  1.0        5          1
src_bytes([358, *)) AND dst_bytes([1695, *)) AND hot([1, 3)) AND num_root([*, 1)) AND dst_host_rerror_rate(0) =>
type_attack(back.)   792        792        1.0        0.033006  0.997481  1.0        5          1
```

```
src_bytes([358, *)) AND hot([1, 3)) AND num_root([*, 1)) AND dst_host_same_srv_rate(1) AND dst_host_rerror_rate(0)
=> type_attack(back.)       792     792     1.0     0.033006  0.997481  1.0     5       1
src_bytes([*, 240)) AND dst_bytes([1695, *)) AND hot([1, 3)) AND dst_host_srv_diff_host_rate([*, 0.01)) =>
type_attack(phf.)   1       1       1.0     0.000042  1.0     1.0     4       1
service(http) AND src_bytes([*, 240)) AND hot([1, 3)) AND count([*, 4)) AND dst_host_srv_diff_host_rate([*, 0.01)) =>
type_attack(phf.)   1       1       1.0     0.000042  1.0     1.0     5       1
dst_bytes([*, 279)) AND srv_diff_host_rate(0) AND dst_host_diff_srv_rate(1) AND dst_host_same_src_port_rate([0.09, *))
AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_rerror_rate(0) => type_attack(nmap.)       92      92
        1.0     0.003834  1.0     1.0     6       1
service(telnet) AND hot([1, 3)) AND num_root([*, 1)) AND dst_host_srv_diff_host_rate([0.03, *)) =>
type_attack(loadmodule.)    1       1       1.0     0.000042  1.0     1.0     4       1
service(telnet) AND hot([1, 3)) AND num_root([*, 1)) AND dst_host_same_srv_rate(1) => type_attack(loadmodule.)
        1       1       1.0     0.000042  1.0     1.0     4       1
service(telnet) AND hot([1, 3)) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([0.03, *))
AND dst_host_rerror_rate(0) => type_attack(loadmodule.)     1       1       1.0     0.000042  1.0     1.0
        5       1
service(telnet) AND hot([1, 3)) AND count([*, 4)) AND dst_host_same_src_port_rate([0.09, *)) AND
dst_host_srv_diff_host_rate([0.03, *)) => type_attack(loadmodule.)      1       1       1.0     0.000042  1.0
        1.0     5       1
service(telnet) AND flag(SF) AND hot([1, 3)) AND num_root([*, 1)) AND count([*, 4)) => type_attack(loadmodule.)
        1       1       1.0     0.000042  1.0     1.0     5       1
service(telnet) AND flag(SF) AND hot([1, 3)) AND dst_host_srv_diff_host_rate([0.03, *)) AND dst_host_rerror_rate(0) =>
type_attack(loadmodule.)    1       1       1.0     0.000042  1.0     1.0     5       1
service(telnet) AND src_bytes([240, 358)) AND dst_host_count([*, 81)) AND dst_host_srv_diff_host_rate([0.03, *)) AND
dst_host_rerror_rate(0) => type_attack(loadmodule.)1       1       1.0     0.000042  1.0     1.0     5
        1
service(telnet) AND hot([1, 3)) AND num_root([*, 1)) AND count([*, 4)) AND dst_host_diff_srv_rate(0) =>
type_attack(loadmodule.)    1       1       1.0     0.000042  1.0     1.0     5       1
service(telnet) AND flag(SF) AND hot([1, 3)) AND num_root([*, 1)) AND dst_host_rerror_rate(0) =>
type_attack(loadmodule.)    1       1       1.0     0.000042  1.0     1.0     5       1
```

APPENDIX G

SEVERAL RULES with COVERAGE 1.0 of SIMPLIFICATION DECISION TABLE

dst_bytes([1695, *)) AND hot([1, 3)) AND num_access_files([1, 2)) => type_attack(phf.) 1 1 1.0
0.000042 1.0 1.0 3 1

logged_in(0) AND count([17, *)) AND srv_count([*, 6)) => type_attack(portsweep.) 18 18 1.0 0.00075
1.0 1.0 3 1

src_bytes([*, 240)) AND root_shell(1) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(phf.) 1 1
1.0 0.000042 1.0 1.0 3 1

flag(S0) AND dst_host_srv_count([*, 255)) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(land.) 1 1
1.0 0.000042 1.0 1.0 3 1

count([17, *)) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_srv_serror_rate(1) => type_attack(portsweep.) 18
18 1.0 0.00075 1.0 1.0 3 1

service(telnet) AND dst_bytes([279, 1695)) AND root_shell(1) => type_attack(loadmodule.) 1 1 1.0
0.000042 1.0 1.0 3 1

num_file_creations([2, *)) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(loadmodule.) 1 1 1.0 0.000042 1.0 1.0 3 1

dst_bytes([279, 1695)) AND root_shell(1) AND dst_host_same_src_port_rate([0.09, *)) => type_attack(loadmodule.) 1
1 1.0 0.000042 1.0 1.0 3 1

dst_bytes([279, 1695)) AND num_compromised([1, 2)) AND dst_host_same_src_port_rate([0.09, *)) => type_attack(loadmodule.)
1 1 1.0 0.000042 1.0 1.0 3 1

src_bytes([*, 240)) AND root_shell(1) AND dst_host_same_src_port_rate([*, 0.01)) => type_attack(phf.) 1 1
1.0 0.000042 1.0 1.0 3 1

src_bytes([240, 358)) AND root_shell(1) AND dst_host_same_src_port_rate([0.09, *)) => type_attack(loadmodule.) 1
1 1.0 0.000042 1.0 1.0 3 1

dst_bytes([279, 1695)) AND hot([1, 3)) AND num_file_creations([2, *)) => type_attack(loadmodule.) 1 1
1.0 0.000042 1.0 1.0 3 1

count([17, *)) AND serror_rate(1) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(portsweep.) 18 18
1.0 0.00075 1.0 1.0 3 1

dst_bytes([279, 1695)) AND root_shell(1) AND num_file_creations([2, *)) => type_attack(loadmodule.) 1 1
1.0 0.000042 1.0 1.0 3 1

service(telnet) AND dst_bytes([279, 1695)) AND dst_host_same_src_port_rate([0.09, *)) AND dst_host_rerror_rate(0) => type_attack(loadmodule.) 1 1 1.0 0.000042 1.0 1.0 4 1

service(telnet) AND dst_bytes([279, 1695)) AND srv_count([*, 6)) AND dst_host_same_src_port_rate([0.09, *)) => type_attack(loadmodule.) 1 1 1.0 0.000042 1.0 1.0 4 1

dst_bytes([279, 1695)) AND num_compromised([1, 2)) AND num_file_creations([2, *)) => type_attack(loadmodule.) 1
1 1.0 0.000042 1.0 1.0 3 1

dst_bytes([*, 279)) AND count([17, *)) AND same_srv_rate(0) => type_attack(portsweep.) 18 18 1.0
0.00075 1.0 1.0 3 1

dst_bytes([279, 1695)) AND num_file_creations([2, *)) AND dst_host_same_src_port_rate([0.09, *)) => type_attack(loadmodule.)
1 1 1.0 0.000042 1.0 1.0 3 1

root_shell(1) AND num_file_creations([*, 1)) AND count([*, 4)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(phf.)
1 1 1.0 0.000042 1.0 1.0 4 1

dst_bytes([1695, *)) AND root_shell(1) AND dst_host_count([255, *)) => type_attack(phf.) 1 1 1.0
0.000042 1.0 1.0 3 1

```
dst_bytes([279, 1695)) AND root_shell(1) AND dst_host_count([*, 81)) => type_attack(loadmodule.)          1           1
        1.0         0.000042    1.0         1.0         3           1
flag(S0) AND dst_host_count([*, 81)) AND dst_host_srv_count([*, 255)) => type_attack(land.)         1           1           1.0
        0.000042    1.0         1.0         3           1
root_shell(1) AND srv_count([*, 6)) AND dst_host_same_src_port_rate([*, 0.01)) => type_attack(phf.)          1           1
        1.0         0.000042    1.0         1.0         3           1
root_shell(1) AND srv_count([*, 6)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(phf.)          1           1
        1.0         0.000042    1.0         1.0         3           1
service(finger) AND flag(S0) => type_attack(land.)          1           1           1.0         0.000042    1.0         1.0         2
        1
dst_bytes([279, 1695)) AND num_compromised([1, 2)) AND num_file_creations([2, *)) => type_attack(loadmodule.)         1
        1           1.0         0.000042    1.0         1.0         3           1
dst_bytes([*, 279)) AND count([17, *)) AND same_srv_rate(0) => type_attack(portsweep.)           18          18          1.0
        0.00075     1.0         1.0         3           1
dst_bytes([279, 1695)) AND num_file_creations([2, *)) AND dst_host_same_src_port_rate([0.09, *)) => type_attack(loadmodule.)
        1           1           1.0         0.000042    1.0         1.0         3           1
root_shell(1) AND num_file_creations([*, 1)) AND count([*, 4)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(phf.)
        1           1           1.0         0.000042    1.0         1.0         4           1
dst_bytes([1695, *)) AND root_shell(1) AND dst_host_count([255, *)) => type_attack(phf.)          1           1           1.0
        0.000042    1.0         1.0         3           1
dst_bytes([279, 1695)) AND root_shell(1) AND dst_host_count([*, 81)) => type_attack(loadmodule.)          1           1
        1.0         0.000042    1.0         1.0         3           1
flag(S0) AND dst_host_count([*, 81)) AND dst_host_srv_count([*, 255)) => type_attack(land.)         1           1           1.0
        0.000042    1.0         1.0         3           1
root_shell(1) AND srv_count([*, 6)) AND dst_host_same_src_port_rate([*, 0.01)) => type_attack(phf.)          1           1
        1.0         0.000042    1.0         1.0         3           1
root_shell(1) AND srv_count([*, 6)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(phf.)          1           1
        1.0         0.000042    1.0         1.0         3           1
service(finger) AND flag(S0) => type_attack(land.)          1           1           1.0         0.000042    1.0         1.0         2
        1
service(ecr_i) AND dst_host_count([255, *)) => type_attack(smurf.)           4971        4971        1.0         0.20716     1.0
        1.0         2           1
dst_bytes([279, 1695)) AND num_compromised([1, 2)) AND dst_host_diff_srv_rate(0) => type_attack(loadmodule.)          1
        1           1.0         0.000042    1.0         1.0         3           1
dst_host_count([255, *)) AND dst_host_diff_srv_rate(0) AND dst_host_srv_serror_rate(1) => type_attack(portsweep.)         18
        18          1.0         0.00075     1.0         1.0         3           1

service(finger) AND serror_rate(1) AND dst_host_same_srv_rate(1) => type_attack(land.)          1           1           1.0
        0.000042    1.0         1.0         3           1
num_root([5, *)) AND dst_host_same_src_port_rate([*, 0.01)) => type_attack(imap.)         1           1           1.0
        0.000042    1.0         1.0         2           1
duration([4, *)) AND num_root([5, *)) => type_attack(imap.)          1           1           1.0         0.000042    1.0         1.0
        2           1
dst_bytes([1695, *)) AND num_root([5, *)) => type_attack(imap.) 1           1           1.0         0.000042    1.0         1.0
        2           1
num_compromised([2, *)) AND num_root([5, *)) => type_attack(imap.)          1           1           1.0         0.000042    1.0
        1.0         2           1
service(telnet)  AND  dst_bytes([279,  1695))  AND  serror_rate(0)  AND  dst_host_same_srv_rate(1)  =>  type_attack(loadmodule.)
        1           1           1.0         0.000042    1.0         1.0         4           1
num_compromised([2, *)) AND dst_host_same_src_port_rate([0.09, *)) => type_attack(buffer_overflow.)          3           3
        1.0         0.000125    1.0         1.0         2           1
```

```
num_compromised([2, *)) AND dst_host_count([*, 81)) => type_attack(buffer_overflow.)          3          3          1.0
          0.000125   1.0          1.0          2          1
num_compromised([2, *)) AND num_file_creations([1, 2)) => type_attack(buffer_overflow.)          3          3          1.0
          0.000125   1.0          1.0          2          1
service(telnet) AND num_file_creations([2, *)) AND dst_host_same_srv_rate(1) => type_attack(loadmodule.)   1          1
          1.0          0.000042   1.0          1.0          3          1
service(telnet) AND num_compromised([1, 2)) AND dst_host_same_srv_rate(1) => type_attack(loadmodule.)   1          1
          1.0          0.000042   1.0          1.0          3          1
service(telnet) AND num_file_creations([2, *)) AND dst_host_same_src_port_rate([0.09, *)) => type_attack(loadmodule.) 1
          1          1.0          0.000042   1.0          1.0          3          1
hot([1, 3)) AND num_access_files([1, 2)) AND dst_host_count([255, *)) => type_attack(phf.)          1          1          1.0
          0.000042   1.0          1.0          3          1
service(telnet)  AND  dst_bytes([279,  1695))  AND  dst_host_same_src_port_rate([0.09,  *))  AND  dst_host_srv_serror_rate(0)  =>
type_attack(loadmodule.)          1          1          1.0          0.000042   1.0          1.0          4          1
duration([4, *)) AND src_bytes([240, 358)) AND hot([1, 3)) AND dst_host_rerror_rate(0) => type_attack(loadmodule.)      1
          1          1.0          0.000042   1.0          1.0          4          1

service(telnet)  AND  src_bytes([240,  358))  AND  dst_host_same_srv_rate(1)  AND  dst_host_same_src_port_rate([0.09,  *))  =>
type_attack(loadmodule.)          1          1          1.0          0.000042   1.0          1.0          4          1
service(finger) AND serror_rate(1) AND dst_host_count([*, 81)) => type_attack(land.)   1          1          1.0
          0.000042   1.0          1.0          3          1
```

APPENDIX H
SEVERAL RULES with ACCURACY 1.0 of SIMPLIFICATION DECISION TABLE

src_bytes([240, 358)) AND count([4, 17)) => type_attack(normal.)	4125	4125	1.0	0.171904	0.229818	1.0	2	1
src_bytes([240, 358)) AND count([17, *)) => type_attack(normal.)	1549	1549	1.0	0.064552	0.0863	1.0	2	1
src_bytes([240, 358)) AND root_shell(0) => type_attack(normal.)	7957	7957	1.0	0.331597	0.443312	1.0	2	1
src_bytes([240, 358)) AND hot([*, 1)) => type_attack(normal.)	7912	7912	1.0	0.329722	0.440805	1.0	2	1
src_bytes([240, 358)) AND hot([3, *)) => type_attack(normal.)	35	35	1.0	0.001459	0.00195	1.0	2	1
src_bytes([*, 240)) AND hot([3, *)) => type_attack(normal.)	11	11	1.0	0.000458	0.000613	1.0	2	1
src_bytes([240, 358)) AND dst_bytes([1695, *)) => type_attack(normal.)	3761	3761	1.0	0.156734	0.209538	1.0	2	1
src_bytes([358, *)) AND dst_bytes([279, 1695)) => type_attack(normal.)	1435	1435	1.0	0.059802	0.079949	1.0	2	1
src_bytes([240, 358)) AND dst_bytes([*, 279)) => type_attack(normal.)	384	384	1.0	0.016003	0.021394	1.0	2	1
src_bytes([240, 358)) AND dst_host_srv_count([255, *)) => type_attack(normal.)	7264	7264	1.0	0.302717	0.404702	1.0	2	1
src_bytes([240, 358)) AND num_compromised([*, 1)) => type_attack(normal.)	7960	7960	1.0	0.331722	0.443479	1.0	2	1
src_bytes([*, 240)) AND num_compromised([1, 2)) => type_attack(ftp_write.)	1	1	1.0	0.000042	0.2	1.0	2	1
src_bytes([240, 358)) AND num_compromised([2, *)) => type_attack(normal.)	1	1	1.0	0.000042	0.000056	1.0	2	1
src_bytes([240, 358)) AND num_compromised([1, 2)) => type_attack(loadmodule.)	1	1	1.0	0.000042	1.0	1.0	2	1
duration([*, 1)) AND src_bytes([240, 358)) => type_attack(normal.)	7867	7867	1.0	0.327846	0.438297	1.0	2	1
duration([1, 4)) AND src_bytes([240, 358)) => type_attack(normal.)	3	3	1.0	0.000125	0.000167	1.0	2	1
service(http) AND src_bytes([240, 358)) => type_attack(normal.)	7875	7875	1.0	0.32818	0.438743	1.0	2	1
service(smtp) AND src_bytes([358, *)) => type_attack(normal.)	1299	1299	1.0	0.054134	0.072372	1.0	2	1
service(ftp_data) AND src_bytes([240, 358)) => type_attack(normal.)	43	43	1.0	0.001792	0.002396	1.0	2	1
service(ecr_i) AND src_bytes([*, 240)) => type_attack(normal.)	55	55	1.0	0.002292	0.003064	1.0	2	1
service(ssh) AND src_bytes([*, 240)) => type_attack(ipsweep.)	1	1	1.0	0.000042	0.006329	1.0	2	1
service(ftp) AND src_bytes([358, *)) => type_attack(normal.)	29	29	1.0	0.001209	0.001616	1.0	2	1

```
service(ftp) AND src_bytes([240, 358)) => type_attack(normal.)   35      35      1.0     0.001459  0.00195   1.0
          2       1
service(ssh) AND src_bytes([358, *)) => type_attack(normal.)     1       1       1.0     0.000042  0.000056  1.0
          2       1
service(ecr_i) AND src_bytes([358, *)) => type_attack(smurf.)    4971    4971    1.0     0.20716   1.0       1.0
          2       1
src_bytes([240, 358)) AND dst_host_same_src_port_rate([0.01, 0.09)) => type_attack(normal.)      3531    3531    1.0
          0.14715   0.196724  1.0     2       1
src_bytes([240, 358)) AND dst_host_same_src_port_rate([*, 0.01)) => type_attack(normal.)         3526    3526    1.0
          0.146941  0.196445  1.0     2       1
src_bytes([240, 358)) AND num_file_creations([*, 1)) => type_attack(normal.)            7961    7961    1.0
          0.331764  0.443534  1.0     2       1
src_bytes([358, *)) AND num_file_creations([2, *)) => type_attack(normal.)   11      11      1.0     0.000458
          0.000613  1.0       2       1
src_bytes([*, 240)) AND num_file_creations([1, 2)) => type_attack(ftp_write.)           1       1       1.0
          0.000042  0.2       1.0     2       1
src_bytes([*, 240)) AND num_file_creations([2, *)) => type_attack(normal.)   1       1       1.0     0.000042
          0.000056  1.0       2       1
src_bytes([240, 358)) AND num_file_creations([2, *)) => type_attack(loadmodule.)        1       1       1.0
          0.000042  1.0       1.0     2       1
dst_host_srv_count([255, *)) AND dst_host_same_src_port_rate([0.01, 0.09)) => type_attack(normal.)       6334    6334
          1.0       0.263961  0.352889  1.0     2       1
service(http) AND dst_host_srv_diff_host_rate([0.03, *)) AND dst_host_rerror_rate(0) => type_attack(normal.) 4878    4878
          1.0       0.203284  0.27177   1.0     3       1
service(http) AND dst_host_srv_diff_host_rate([0.01, 0.03)) AND dst_host_rerror_rate(0) => type_attack(normal.)     3811
          3811      1.0       0.158818  0.212324  1.0     3       1
service(ftp_data) AND dst_host_srv_diff_host_rate([0.03, *)) AND dst_host_rerror_rate(0) => type_attack(normal.)    220
          220       1.0       0.009168  0.012257  1.0     3       1
service(smtp) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_rerror_rate(0) => type_attack(normal.)        336
          336       1.0       0.014002  0.01872   1.0     3       1
service(http) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_rerror_rate(1) => type_attack(normal.) 13      13
          1.0       0.000542  0.000724  1.0     3       1
service(ftp_data) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_rerror_rate(0) => type_attack(normal.)    178
          178       1.0       0.007418  0.009917  1.0     3       1
service(http) AND dst_host_diff_srv_rate(0) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(normal.)      5483
          5483      1.0       0.228496  0.305477  1.0     3       1
service(telnet) AND dst_host_diff_srv_rate(1) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(normal.)    3
          3         1.0       0.000125  0.000167  1.0     3       1
service(http) AND dst_host_diff_srv_rate(0) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)   3814
          3814      1.0       0.158943  0.212491  1.0     3       1
service(ftp_data) AND dst_host_diff_srv_rate(0) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(normal.)  174
          174       1.0       0.007251  0.009694  1.0     3       1
service(domain_u) AND dst_host_diff_srv_rate(0) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(normal.)  37
          37        1.0       0.001542  0.002061  1.0     3       1
service(http) AND dst_host_srv_diff_host_rate([0.03, *)) AND dst_host_srv_rerror_rate(0) => type_attack(normal.)    4843
          4843      1.0       0.201825  0.26982   1.0     3       1
service(http) AND dst_host_srv_diff_host_rate([0.01, 0.03)) AND dst_host_srv_rerror_rate(0) => type_attack(normal.) 3814
          3814      1.0       0.158943  0.212491  1.0     3       1
service(http) AND dst_host_srv_diff_host_rate([*, 0.01)) AND dst_host_srv_rerror_rate(1) => type_attack(normal.)    13
          13        1.0       0.000542  0.000724  1.0     3       1
```

```
service(domain_u) AND dst_host_srv_diff_host_rate([0.03, *)) AND dst_host_srv_rerror_rate(0) => type_attack(normal.)  38
          38        1.0       0.001584  0.002117  1.0       3         1
service(http) AND count([4, 17)) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(normal.)          2436      2436
          1.0       0.101517  0.135718  1.0       3         1
service(http) AND count([17, *)) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(normal.)          518       518
          1.0       0.021587  0.02886   1.0       3         1
service(domain_u) AND count([*, 4)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(normal.)       200       200
          1.0       0.008335  0.011143  1.0       3         1
service(http) AND count([4, 17)) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)       2047      2047
          1.0       0.085306  0.114045  1.0       3         1
service(http) AND count([*, 4)) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)        1152      1152
          1.0       0.048008  0.064182  1.0       3         1
service(finger) AND count([*, 4)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(normal.)         30        30
          1.0       0.00125   0.001671  1.0       3         1
service(ftp_data) AND count([*, 4)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(normal.)       100       100
          1.0       0.004167  0.005571  1.0       3         1
service(eco_i) AND count([*, 4)) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(ipsweep.)         82        82
          1.0       0.003417  0.518987  1.0       3         1
service(smtp) AND count([*, 4)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(normal.)           341       341
          1.0       0.014211  0.018998  1.0       3         1
dst_host_diff_srv_rate(0)   AND   dst_host_same_src_port_rate([0.01,   0.09))   AND   dst_host_srv_diff_host_rate([0.03,   *))   =>
type_attack(normal.)  3873      3873      1.0       0.161402  0.215778  1.0       3         1
dst_host_diff_srv_rate(0)   AND   dst_host_same_src_port_rate([0.01,   0.09))   AND   dst_host_srv_diff_host_rate([0.01,   0.03))   =>
type_attack(normal.)  3548      3548      1.0       0.147858  0.197671  1.0       3         1
dst_host_diff_srv_rate(0)   AND   dst_host_same_src_port_rate([*,   0.01))   AND   dst_host_srv_diff_host_rate([0.01,   0.03))   =>
type_attack(normal.)  584       584       1.0       0.024337  0.032537  1.0       3         1
```

APPENDIX I
SEVERAL RULES with LENGTH 2 of SIMPLIFICATION DECISION TABLE

```
count([4, 17)) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(normal.)      2514      2514      1.0
          0.104767   0.140064   1.0        2        1
count([4, 17)) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)   2137      2137      1.0
          0.089057   0.11906    1.0        2        1
count([17, *)) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(normal.)      529       529       1.0
          0.022045   0.029472   1.0        2        1
count([17, *)) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)   632       632       1.0
          0.026338   0.035211   1.0        2        1
src_bytes([240, 358)) AND count([4, 17)) => type_attack(normal.) 4125      4125      1.0       0.171904   0.229818   1.0
          2          1
src_bytes([240, 358)) AND count([17, *)) => type_attack(normal.) 1549      1549      1.0       0.064552   0.0863     1.0
          2          1
src_bytes([240, 358)) AND dst_bytes([1695, *)) => type_attack(normal.)      3761      3761      1.0       0.156734
          0.209538   1.0        2          1
src_bytes([358, *)) AND dst_bytes([279, 1695)) => type_attack(normal.)      1435      1435      1.0       0.059802
          0.079949   1.0        2          1
src_bytes([240, 358)) AND dst_bytes([*, 279)) => type_attack(normal.)       384       384       1.0       0.016003
          0.021394   1.0        2          1
service(http) AND src_bytes([240, 358)) => type_attack(normal.)  7875      7875      1.0       0.32818    0.438743   1.0
          2          1
service(smtp) AND src_bytes([358, *)) => type_attack(normal.)    1299      1299      1.0       0.054134   0.072372   1.0
          2          1
service(ftp_data) AND src_bytes([240, 358)) => type_attack(normal.)         43        43        1.0       0.001792
          0.002396   1.0        2          1
service(ecr_i) AND src_bytes([*, 240)) => type_attack(normal.)   55        55        1.0       0.002292   0.003064   1.0
          2          1
service(ssh) AND src_bytes([*, 240)) => type_attack(ipsweep.)    1         1         1.0       0.000042   0.006329   1.0
          2          1
service(ftp) AND src_bytes([358, *)) => type_attack(normal.)     29        29        1.0       0.001209   0.001616   1.0
          2          1
service(ftp) AND src_bytes([240, 358)) => type_attack(normal.)   35        35        1.0       0.001459   0.00195    1.0
          2          1
service(ssh) AND src_bytes([358, *)) => type_attack(normal.)     1         1         1.0       0.000042   0.000056   1.0
          2          1
service(finger) AND src_bytes([358, *)) => type_attack(portsweep.)         1         1         1.0       0.000042
          0.055556   1.0        2          1
service(ecr_i) AND src_bytes([358, *)) => type_attack(smurf.)    4971      4971      1.0       0.20716    1.0        1.0
          2          1
src_bytes([240, 358)) AND dst_host_same_src_port_rate([0.01, 0.09)) => type_attack(normal.)      3531      3531      1.0
          0.14715    0.196724   1.0        2        1
src_bytes([240, 358)) AND dst_host_same_src_port_rate([*, 0.01)) => type_attack(normal.)         3526      3526      1.0
          0.146941   0.196445   1.0        2        1
src_bytes([240, 358)) AND hot([*, 1)) => type_attack(normal.)    7912      7912      1.0       0.329722   0.440805   1.0
          2          1
src_bytes([240, 358)) AND hot([3, *)) => type_attack(normal.)    35        35        1.0       0.001459   0.00195    1.0
          2          1
```

```
src_bytes([*, 240)) AND hot([3, *)) => type_attack(normal.)        11        11        1.0        0.000458  0.000613  1.0
        2        1
dst_host_same_src_port_rate([0.01, 0.09)) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)        3556
        3556        1.0        0.148191  0.198117  1.0        2        1
dst_host_same_src_port_rate([*, 0.01)) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.) 584        584
        1.0        0.024337  0.032537  1.0        2        1
dst_host_same_src_port_rate([*, 0.01)) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(normal.)     29        29
        1.0        0.001209  0.001616  1.0        2        1
dst_bytes([1695, *)) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)        1859        1859        1.0
        0.077471  0.103571  1.0        2        1
dst_bytes([279, 1695)) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)        2662        2662        1.0
        0.110935  0.148309  1.0        2        1
src_bytes([240, 358)) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)        1984        1984        1.0
        0.08268   0.110535  1.0        2        1
src_bytes([240, 358)) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(normal.)        3414        3414        1.0
        0.142274  0.190206  1.0        2        1
service(http) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)        3814        3814        1.0
        0.158943  0.212491  1.0        2        1
service(smtp) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(normal.)        341        341        1.0
        0.014211  0.018998  1.0        2        1
service(ftp_data) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(normal.)        180        180        1.0
        0.007501  0.010028  1.0        2        1
service(private) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(ipsweep.)        41        41        1.0
        0.001709  0.259494  1.0        2        1
service(smtp) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)        854        854        1.0
        0.035589  0.047579  1.0        2        1
service(time) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(ipsweep.)        1        1        1.0
        0.000042  0.006329  1.0        2        1
service(ssh) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(ipsweep.)        1        1        1.0
        0.000042  0.006329  1.0        2        1
service(ssh) AND dst_host_srv_diff_host_rate([*, 0.01)) => type_attack(normal.)        1        1        1.0
        0.000042  0.000056  1.0        2        1
service(domain_u) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)        47        47        1.0
        0.001959  0.002619  1.0        2        1
service(pop_3) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(normal.)        11        11        1.0
        0.000458  0.000613  1.0        2        1
service(domain_u) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(normal.)  38        38        1.0
        0.001584  0.002117  1.0        2        1
service(eco_i) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(ipsweep.)        82        82        1.0
        0.003417  0.518987  1.0        2        1
service(ecr_i) AND dst_host_srv_diff_host_rate([0.03, *)) => type_attack(normal.)        6        6        1.0        0.00025
        0.000334  1.0        2        1
service(ecr_i) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)        2        2        1.0
        0.000083  0.000111  1.0        2        1
dst_bytes([279, 1695)) AND count([4, 17)) => type_attack(normal.)        3661        3661        1.0        0.152567
        0.203967  1.0        2        1
dst_bytes([279, 1695)) AND count([17, *)) => type_attack(normal.)        1496        1496        1.0        0.062344
        0.083347  1.0        2        1
service(http) AND dst_bytes([279, 1695)) => type_attack(normal.) 6681        6681        1.0        0.278421  0.372221  1.0
        2        1
```

```
service(smtp) AND dst_bytes([279, 1695)) => type_attack(normal.)            1287        1287        1.0         0.053634
            0.071703   1.0          2            1
service(smtp) AND dst_bytes([1695, *)) => type_attack(normal.)  8           8           1.0         0.000333   0.000446   1.0
            2            1
service(ftp) AND dst_bytes([*, 279)) => type_attack(ipsweep.)   1           1           1.0         0.000042   0.006329   1.0
            2            1
service(ssh) AND dst_bytes([*, 279)) => type_attack(ipsweep.)   1           1           1.0         0.000042   0.006329   1.0
            2            1
service(ftp) AND dst_bytes([1695, *)) => type_attack(normal.)   26          26          1.0         0.001084   0.001449   1.0
            2            1
service(ssh) AND dst_bytes([1695, *)) => type_attack(normal.)   1           1           1.0         0.000042   0.000056   1.0
            2            1
service(finger) AND dst_bytes([279, 1695)) => type_attack(normal.)          7           7           1.0         0.000292   0.00039
            1.0          2            1
service(pop_3) AND dst_bytes([1695, *)) => type_attack(normal.) 1           1           1.0         0.000042   0.000056   1.0
            2            1
srv_count([6, 23)) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)         2080        2080        1.0
            0.086681   0.115884   1.0          2            1
srv_count([23, *)) AND dst_host_srv_diff_host_rate([0.01, 0.03)) => type_attack(normal.)         612         612         1.0
            0.025504   0.034097   1.0          2            1
```

APPENDIX J
LIST OF PUBLICATION

List of Journal Publications;

Noor, S. S. & Rohani, A. B., (2016) Rough Set Theory Discretization: Equal Frequency Binning, Entropy, MDL and Semi Naives Algorithm of Intrusion Detection System. *Journal of Intelligent Computing, 8*(3), 91-97.

Noor, S. S. & Rohani, A. B., (2016) Rough Set Theory Significant Reduct and Rules of Intrusion Detection System. *Journal of Information Security Research, 8*(1), 17-25.

Noor, S. S. & Rohani, A. B., (2016) Rough Set Discretize Classification of Intrusion Detection System. *Journal of Engineering and Applied Science, 11*(3), 488-496.

Noor, S. S. & Rohani, A. B., (2012) A Review on Soft Computing Technique in Intrusion Detection System. *International Journal of Computer, Electrical, Automation, Control and Information Engineering, 6*(12).

List of Conference Proceeding;

Noor, S. S. & Rohani, A. B., (2016) Rough Set Theory Discretization: Equal Frequency Binning, Entropy, MDL and Semi Naives Algorithm of Intrusion Detection System. *Advances in Digital Technologies – Proceeding of the 7th International Conference on Application of Digital Information and Web Technologies.*

Noor, S. S. & Rohani, A. B., (2016) Rough Set Theory Significant Reduct and Rules of Intrusion Detection System. *Advances in Digital Technologies – Proceeding of the 7th International Conference on Application of Digital Information and Web Technologies.*

Noor, S. S. & Rohani, A. B., (2013) Artificial Intelligent Based Technique in Intrusion Detection System: A Review. *Procceding of the International Conference on Engineering & Technology.*

Noor, S. S. & Rohani, A. B., (2012) A Review on Soft Computing Technique in Intrusion Detection System. *Procceding of the World Academy of Science, Engineering and Technology.*