ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

TELECOMMUNICATION ENGINEERING GRADUATE PROGRAM

COMPARISON OF MACHINE LEARNING TECHNIQUES FOR INTRUSION DETECTION SYSTEM

A thesis submitted to Addis Ababa University, Addis Ababa Institute of Technology School of Electrical and Computer Engineering, in partial fulfillment of the requirements for the degree of Masters of Science in Telecommunications Engineering (Telecommunication Information Systems Track)

By

Kelem Birhane

Advisor: Henock Mulugeta (PhD)

OCTOBER 2018

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

TELECOMMUNICATION ENGINEERING GRADUATE PROGRAM

COMPARISON OF MACHINE LEARNING TECHNIQUES FOR INTRUSION DETECTION SYSTEM

By

Kelem Birhane

APPROVAL BY BOARD OF EXAMINERS

Henock Mulugeta (PhD)

_____	_____
Advisor	Signature
_____	_____
Internal Examiner	Signature
_____	_____
External Examiner	Signature

DECLARATION

I declare that the thesis is my original work and has not been presented for a degree in any other university.

Kelem Birhane

OCTOBER 2018

Dedication

I am dedicating this thesis to my father (Ato Birhane Gebreegziabher) for his scarification for me to reach here and his courage and devotion to realize my dream.

Acknowledgement

First and foremost, my thanks would like go to the Almighty God and His Mother Saint-Merry. Secondly, I would like to express my sincere gratitude to my advisor Dr. Henock Mulugeta for the continuous support of my thesis study, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis, furthermore his insightful comments and encouragement which added great value for my future career.

Besides my advisor, I would like to thank for ethio telecom for letting me study my masters with full sponsor and I am also pleased to thank for our lecturers for their contribution in the special arena for the success of my study and for their selfless support.

I would like to extend my gratitude to my friend Tigabu Dagne. I greatly value My Coo supervisor Tigabu for his valuable support during my thesis work. I would like to thank my classmate Abera Resom too. He is selfless and generous guy and was my courage during my study.

Finally, I must express my very profound gratitude to my family specially Letish for providing me with unfailing support and continuous encouragement throughout my years of study. Thank you

Abstract

The rapid growth in the ubiquity and sophistication of Information Communication Technology (ICT) and the emergence of new networking paradigms such as Cloud Computing (CC), and Internet of Things (IoT) have made vital changes in the globe. Computer network security is one of the most critical issue as attackers are also evolving dynamically. There should be a mechanism that fill the security vulnerability. One of the promising technique to ensure computer network security is the use of hybrid machine learning (ML) techniques which automate the process of intrusion detection in computer networks.

In this research, six hybrid ML models were developed based on the Knowledge Discovery in Database (KDD) process model. The dataset used in this study has been taken from University of New Brunswick Institute (Canada Institute of Cyber Security). After selecting the dataset, preprocessing techniques such as filling missing records, reduce dimension, selecting the most relevant features, and finally normalize the dataset input using features scaling are performed. The hybrid ML models for intrusion detection systems (IDS) are implemented using Python programming language.

In this work, a total of 274208 dataset records are used for the ML models evaluation. Out of this, 191945 datasets are used for training and a separate 82263 records are used as a testing set. The decision tree (DT) and neural network (NN) algorithms as supervised and K-means algorithm as unsupervised algorithms are applied in both without feature selection and with feature selection.

The principal component analysis and decision tree (PCA-DT) model showed the best results in all performance parameters. The model has a prediction accuracy of 99.89% and

the lowest false positive rate of 0.027%. Results confirm the effectiveness of our proposed methods.

Keywords: Intrusion Detection, Machine learning algorithms, Computer Security

Table of Contents

List of Figures

List of Tables

List of Acronyms

ADTree	Alternating Decision Tree
ANN	Artificial Neural Network
CICIDS	Canadian Institute for Cyber security intrusion detection system
CSV	Comma Separated Values
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denials of Service
DOS	Denials of Service
DT	Decision Tree
FCM	Fuzzy C-Mean
FI	feature Importance
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
KDD	Knowledge discovery in Database
KNN	K-Nearest Neighbor
MIT	Massachusetts Institute of Technology
ML	Machine Learning
NIDS	Network Intrusion Detection System
NN	Neural Network
NSL-KDD	Network Simulation Language Knowledge Discovery in Database
R2L	Remote to Local
ROC	Receiver Operating Characteristics
SSH	Secure Socket Shell
SVM	Support Vector Machine
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate

Chapter 1

Introduction

1.1 Background

1.1.1 Computer and Network Security

The rapid growth in the ubiquity and sophistication of Information Communication Technology (ICT) and also the emergence of new networking paradigms such as Cloud Computing (CC), Internet of Things (IoT) and big data has brought vital transformation in business and people's day to day life. ICT has become an indispensable platform for our life, and became a bridge of communication. It is a tool that keeps individuals and enterprises highly relayed on it for efficient execution of business and services [1, 2].

Today, every aspect of our life is influenced by networked computer systems. Being it is developed beyond our imagination, it has introduced new security challenges that need a new mechanism to assure network security requirements. Networked computer systems become the target of cyber-attacks to compromise the security criteria and meet their target motivation [1, 3].

Security in the context of computer networks refers to the protection of information system with the objective of ensuring the network security requirements. ITU-T Recommendation X.805 Security architecture has defined eight security dimensions which are applied by service providers or enterprises to ensure security of their ICT systems [4]. The compromising of these security criteria will cause for unauthorized access, modification or denial of service of the computer system might occur. Figure 1.1

describes the objectives of applying the security dimensions to computer networks. The security dimensions are defined to address end-to-end security of networked applications i.e. privacy, data confidentiality, authentication, integrity, non-repudiation, access control, communication security, and availability [4].

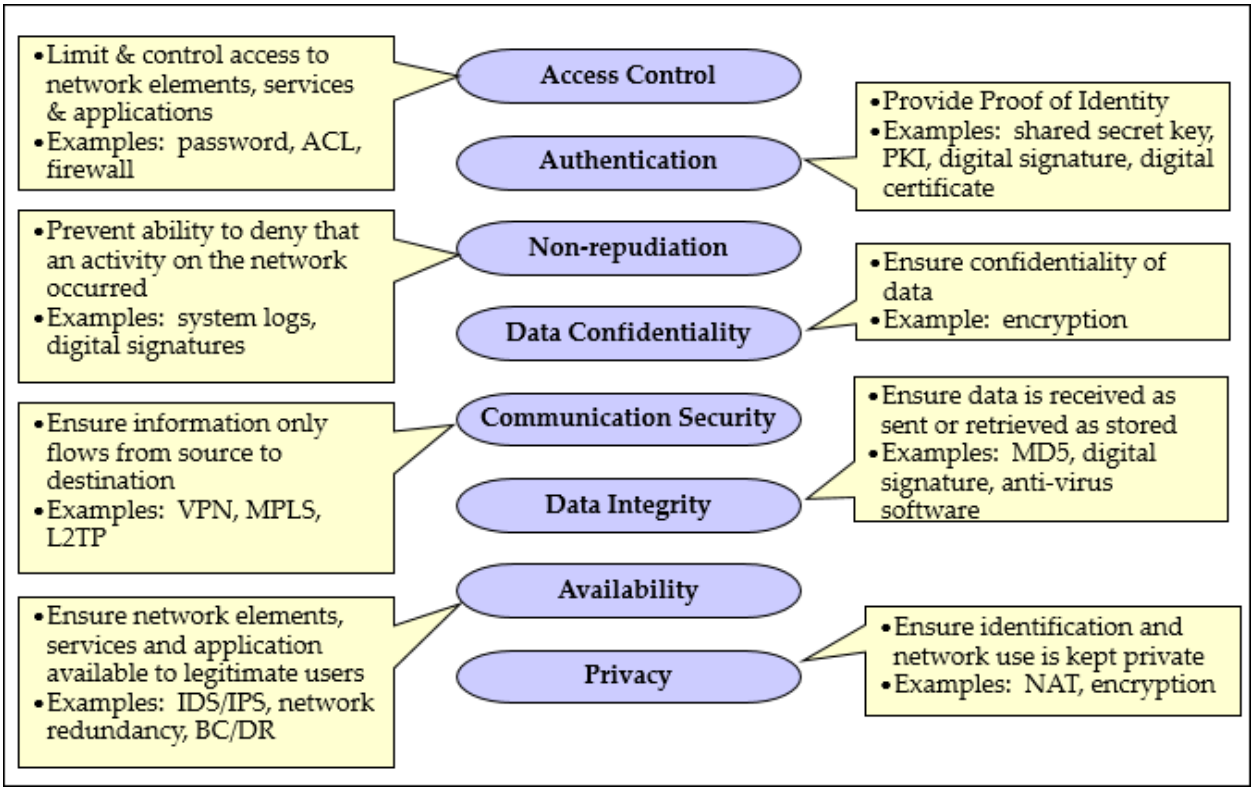


Figure 1.1: Security Dimensions applied to network infrastructure [4]

The activities that attempt to violate the security requirements are known as attacks. According to Juniper network whitepaper [3], attackers knowledge and understanding of technology have increased which is any enterprise network can be attacked easily. Figure 1.2 illustrates the frequency and sophistication of cyber-attacks. The knowledge an attacker needs to know about enterprise network in order to launch a sophisticated attack is decreasing. This means sophisticated attacks are growing more severe each day.

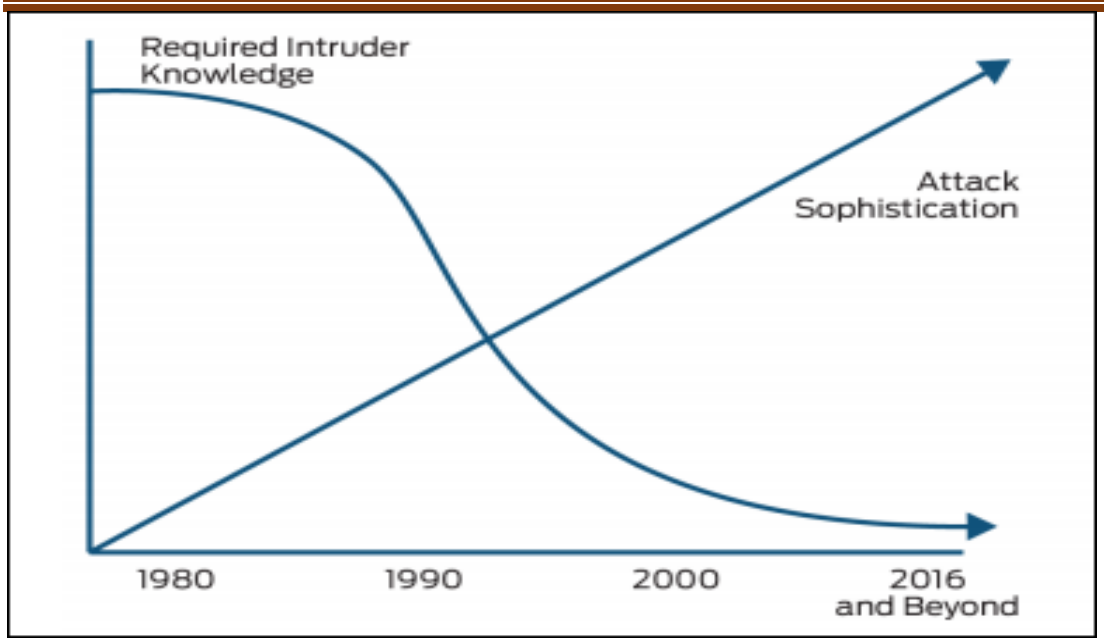


Figure 1.2: Attack Sophistication vs Intruder Technical Knowledge [3]

The increasing interconnectivity of infrastructure, and sharing of sensitive information on the Internet are few of the reasons that the landscape of cyber-attack significantly increasing dynamically and the potential damage that can be caused by launching attacks is becoming obvious. According to the ITU 2017 global cyber security index report, in 2016, one percent of all emails sent were essentially malicious attacks, which is the highest rate in recent years [5].

According to the report in [6], cyber-attack might become the greatest threat for enterprise and individuals. It also indicated that security events have been declined, but damage by cybercrime has become significant and annual cyber-attack damages costs will hit $6T annually, and cyber-defense spending will exceed $1B by 2021.

1.1.2 Intrusion and Intrusion Detection System (IDS)

Intrusion is defined as malicious, externally or internally tempted and is the act of using a computer system or resources without the requisite privileges. It could cause loss of availability, integrity and confidentiality of networked computer systems. Network

intrusions are classified in to four categories based on the defense advanced research projects agency (DARPA) intrusion detection evaluation plan: Denials of Service (DoS), Probing, Use to Root (U2R), and Remote to Local (R2L) attacks [7]. The four categories of attack types are described in Table 1.1.

Table 1.1: Intrusion Categories Based on KDD98 [7]

Attack Category	Description
Denials of Service (DoS)	A denial-of-service (DoS) is any type of attack where the attackers attempt to prevent legitimate users from accessing the service. In a DoS attack, the attacker usually sends excessive messages asking the network or server to authenticate requests. Common forms of denial of service attacks are buffer overflow, ping of death (PoD), TCP SYN, smurf, neptune, and teardrop
Probing	A probing is an attack in which the hacker scans a machine or a networking device in order to determine weaknesses or vulnerabilities that may later be exploited so as to compromise the system. Some common probe attacks are: ▪ IPsweep:- scans the network for a service on a specificc port ▪ Portsweep:- scans through many ports to determine which services are supported on a single host
User to Root (U2R)	Is a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system.

Attack Category	Description
Remote to Local (R2L)	Occurs when an attacker who has the ability to send packets to Computer machine over a network to exploit some vulnerability to gain local access as a user of that machine.

Intrusion detection is defined as the process of monitoring the events occurring in a computer system and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity availability or bypass the security mechanism of networked computer systems. It identifies unauthorized use and misuse of computer systems by both system insiders and external intruders[8]. IDS detect intrusions by analyzing information about user activity from sources such as audit records, log files, system tables, and network traffic summaries [9].

From a system architecture perspective, IDS has various components: audit data processor, knowledge base, decision engine, alarm generation and responses. Figure 1.3 shows the generic architecture of intrusion detection system [1].

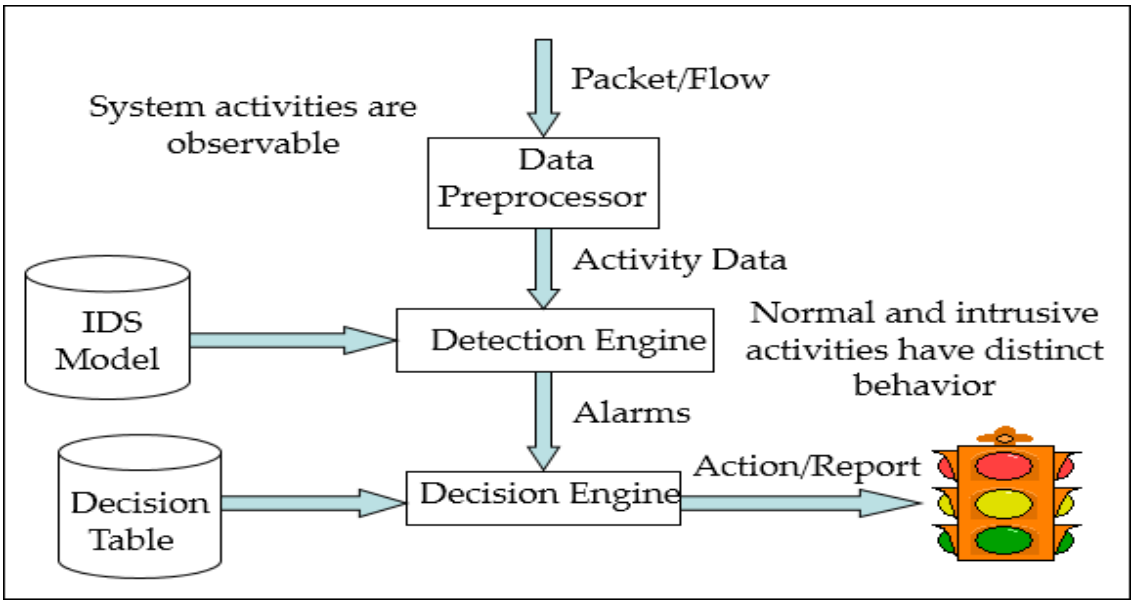


Figure 1.3: Intrusion Detection System architecture [1]

- **Data Preprocessor**- is responsible for collecting and providing the data that will be used by next component to make decisions. Then the collected packet or flow level data will be preprocessed before sending to the detection engine.
- **Detection Engine**- The intrusion detector is the core component which analyzes the audit patterns to detect attacks. It uses the detection model using data mining, pattern matching, soft computing, machine learning and various statistical techniques used as an intrusion detector.
- **Decision Table-** used to describe the normal and abnormal user behavior. It is the database of the audit information, attacks, and events that are going to happen on the system.
- **Decision Engine-** The decision engine controls the reaction mechanism and determines how to respond based on the policies which are stored in the decision table. The system may raise an alarm and report to administrator based on the rules or policy in the decision table**.**
- **IDS Model-** IDS model is based on the hypothesis that security violations can be detected by monitoring a system's audit records for abnormal patterns of system usage. It is an independent system which has profiles which represent the benign and anomalous behavior and rules for acquiring knowledge about the normal behavior from audit records and for detecting anomalous behavior.

1.1.3 Intrusion Detection Approaches

Intrusion detection has been studied for many years and IDS tools have been developed [1]. There are several approaches on the implementation of IDS. Based on its deployment, IDS can act either as a host based or as a network based IDS. The host based IDS monitors and analyzes the internals of a computing system where as the network based IDS

analyzes network traffic at all layers of the TCP/IP suite and detects intrusions in network traffic flows from both directions. It is also categorized in to anomaly and signature based from the perspective of the detection approach applied [1, 8].

Signature based detection approach is an IDS technique in which detection of intrusion is based on the signature of known attacks tuned in the database. In misuse detection approach, it identifies suspicious data by comparing new instances with the stored signatures or patterns of attacks in the database. It is efficient and comparably low false alarm rate IDS technique but it cannot detect emerging or new attack types [1].

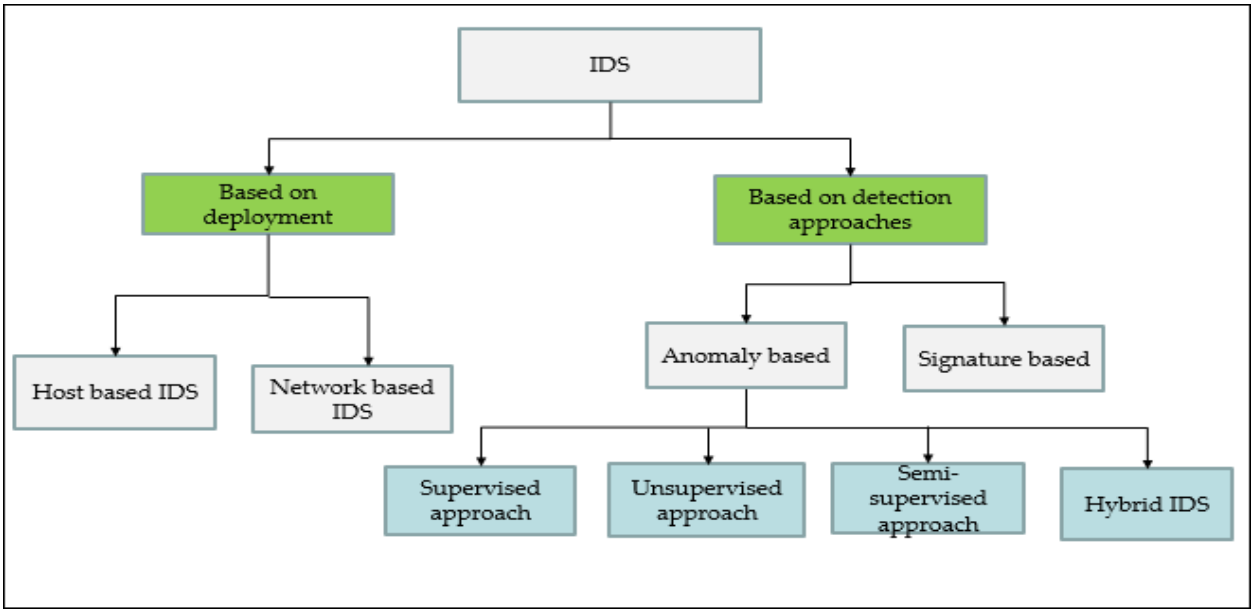


Figure 1.4: Classification of intrusion detection techniques [1]

Anomaly based detection approach detects the behavior of a system deviations from the normal activity as anomalies. Baseline of normal data in network data is defined, in which the IDS has knowledge of normal behavior it monitors new instances. The new instances are compared with the baseline, if there is any deviation from baseline, data is notified as intrusion. Its main advantage is it can detect new attack types but generate high false

alarms rates. The anomaly detection approach can work basically in four modes, as shown in Figure 1.4 in the above [1].

- **Supervised approach**- this technique needs a training dataset which has labeled instances for normal as well as anomaly classes. It can detect known attacks only which the foremost drawback of this approach.
- **Semi-supervised approach-** it is an approach that operate in a semi-supervised mode assuming that the training dataset has labeled instances for only the normal class.
- **Unsupervised Learning-** When performing unsupervised learning, the machine is presented with totally unlabeled data. It is asked to discover the intrinsic patterns that underlies the data, such as a clustering structure, a low-dimensional manifold, or a sparse tree and graph. In this method clustering and dimension reduction techniques can be applied depending on the type the problem.
- **Hybrid approach-** A hybrid intrusion detection method combines supervised and unsupervised network intrusion detection methods. Supervised methods have the advantage of being able to detect known attacks with high accuracy and low false alarms. On the other hand, an unsupervised method has the ability to detect novel attacks. Thus, a hybrid intrusion detection method is capable of detecting both known as well as unknown attacks with high accuracy. Figure 4 shows the classification of intrusion detection approaches.

1.2 Statement of the Problem

With the current advancement in network technology, the emergence of new networking paradigms which are Cloud Computing (CC) and Internet of Things (IoT), cyber-attacks are also evolving and adapting their techniques at a faster pace than defenders. Security is paramount importance which enterprises should also consider adopting advanced security technologies to complement threat detection and prevention [2, 10].

This research is driven based on the current ethio telecom security experience. As ethio telecom is an enterprise which has enormous critical IT systems in its network, improving its security system is crucial. Interview was conducted with ethio telecom network security experts to collect data about the current IDS implemented. Currently, ethio telecom is using signature based intrusion detection system. Signature based intrusion detection compares signatures against observed events to identify intrusion incidents. This method requires prior knowledge of attack signatures and hence cannot identify new categories of attack for which signatures have not been available in database. It needs manual update of the signature database which is tedious, expensive as well as error prone job. As cyber-attack is evolving dynamically, signature based intrusion detection system will not sufficient to defend from internal and external intruders.

Machine learning techniques provide a promising solution for improving intrusion detection system (IDS). Machine learning techniques are classified in to supervised and unsupervised learning techniques. Supervised learning needs a training dataset with labeled instances for normal as well as anomaly classes, whereas in unsupervised learning, the algorithm directly learns patterns from the data, without any human intervention. Developing IDS model with combination of supervised and unsupervised learning algorithms called hybrid intrusion detection system has become an important solution to detect existing attacks and emerging attacks. Using combination supervised

and unsupervised machine learning algorithms for intrusion detection can improve its overall performance as the weaknesses of one algorithm might be complemented by the second one.

Security system of enterprise network should be improved in line with technology advancement to enhance network security defenses. This research has attempted to build a predictive hybrid ML models for intrusion detection using a new benchmark dataset Canadian Institute for Cyber security intrusion detection system (CIC-IDS2017) for performance evaluation. Applying a new dataset to meet the current significant advances in internet traffic diversity and emerging attacks types is mandatory.

1.3 Objective of the Study

1.3.1 General Objective

The general objective of this study is to build ML model for intrusion detection system (IDS) using a hybrid approach that will enhance the computer network security system.

1.3.2 Specific Objectives

The specific objectives of this research study are:

- To study different classification of intrusion detection techniques
- To process the dataset required and identify the best features
- To build ML model using hybrid machine learning approach
- To conduct training and testing of the predictive models using the new CIC-IDS2017 benchmark dataset
- To compare the ML models using evaluation metrics
- To interpret and analyze the results of the selected model

1.4 Significance of the Study

The emergence of new networking paradigms and dynamic change of digitization has introduced new security challenges due to heterogeneous and ubiquitous nature of those technology will demand new techniques to guarantee the security requirements of enterprise network [2]. Due to this fact deploying only signature based IDS as a security tool is not enough to meet the current enterprises network security requirement.

In this study, we have proposed hybrid ML model which combines supervised and unsupervised approaches which is promising solution to detect known attacks which have already stored signatures or labeled one, and new emerging attacks those which have not label. This research will enhance the effectiveness and efficiency of network intrusion detection system by proposing efficient feature selection and classifier techniques.

The big data derives to develop variety of the machine learning algorithms suitable for cyber security problems. Many researches have been made in this area and many authors applied single algorithm (supervised or unsupervised approaches) and semi-supervised approaches to address the intrusion detection problem. From the literature view, we have observed that most of the IDS researches are used the KDD and improved version NSL-KDD for performance evaluation. This dataset could not provide the required accuracy for evaluation as the internet traffic and attack types are dynamically changed and diversified. This study we have conducted using the new CIC-IDS2017 benchmark dataset for the performance evaluation of IDS models which can simulate the current traffic behavior.

There are three main contributions in this thesis work. First, the IDS models are evaluated using the new CIC-IDS2017 benchmark dataset. It motivates the benchmark dataset that

can handle new emerging cyber-attacks and may result less false positive rate and high accuracy as it can simulate the current internet traffic behavior.

Second, we take our work further and implement the system in a Python language which is emerging language for data science and apply CIC-IDS2017 benchmark dataset for experiment and measure its performance. This encourages further research to improve the proposed IDS.

Finally, this research will contribute its part for ethio telecom and similar enterprises, because it can create awareness of applying big data to solve problems or get competitive advantage as enterprises are moving to data-driven decision. In addition to this, it will add knowledge for professionals who want to advance their area of study in network security domain area.

1.5 Research Methodology

The methodologies to be used in conducting this research are described as follows.

1.5.1 Literature Review

The researcher has made review of different related literatures i.e. books, journal articles, Conference proceeding papers, and the Internet in order to have detailed understanding on the present research trend in the study area. In addition to this, different techniques and tools which are relevant for the current research identified, studied and adopted for this study. The detailed reviewed literatures of this study are presented in section 1.6.

1.5.2 Data Collection

From the literature review, new intrusion detection evaluation dataset is identified. The dataset is CIC-IDS 2017 is prepared by University of New Brunswick institute of cyber security. This is new introduced benchmark dataset for effective and robust evaluation

of machine learning based intrusion detection system. This study is conducted using the new CIC-IDS2017 benchmark dataset for the performance evaluation of IDS models.

1.5.3 Data Processing and ML Creation

As we are in the age of digital information, the databases of modern science are so immense which is difficult to analysis and discover new knowledge from it using manually. Researchers have begun to search for ways to automate its analysis as traditional techniques for analysis and visualization of the data are not possible. A new generation of computational techniques and tools is required to support the extraction of useful knowledge from the rapidly growing volumes of data. These techniques and tools are the subject of the emerging field of knowledge discovery in databases (KDD) [11]. The machine learning process is conducted using the KDD process framework model which is the whole process of changing low level data into high level knowledge. It is automated discovery of patterns and relationships in large databases and data mining or machine learning is one of the core steps in the KDD process framework illustrated in Figure 1.5 [12, 13].

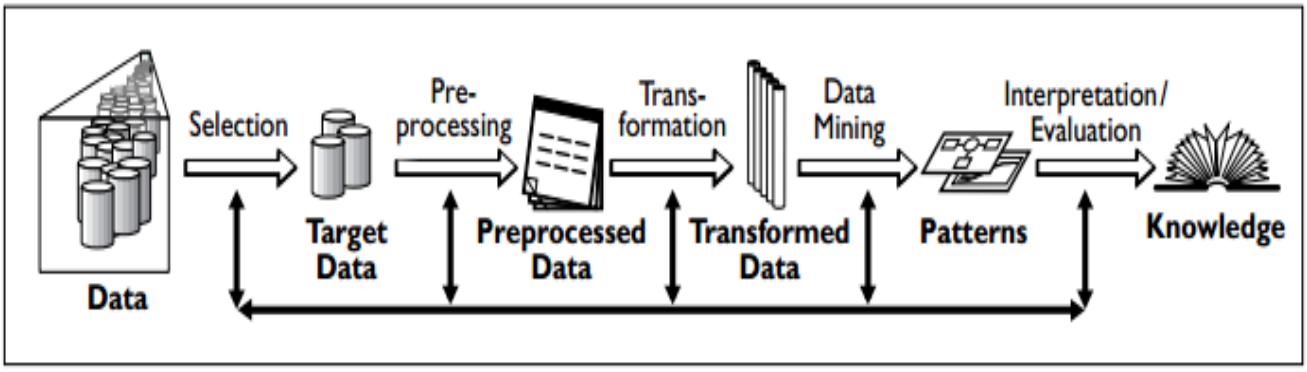


Figure 1.5: KDD process model [11]

The KDD process model is described in these five major steps briefly below [11]

- **Data Selection-** Creating a target dataset includes selecting a dataset or focusing on a subset of variables or data samples on which discovery is to be performed.
- **Data Preprocessing**- Data cleaning and preprocessing includes basic operations, such as removing noise or outliers if appropriate, collecting the necessary information to model or account for noise, deciding on strategies for handling missing data fields, and accounting for time sequence information and known changes.
- **Data Transformation**- The data transformation step includes finding useful features to represent the data, depending on the goal of the task, using dimensionality reduction or transformation methods to reduce features with no effect in the model performance
- **Choosing ML Algorithms and Approaches-** In this step the ML algorithms and the approaches (supervised, semi-supervised or hybrid) used for the thesis are decided. In this study, the hybrid machine learning approach is used to build ML models.
- **ML Model Evaluation-** It is the final step in the KDD process framework. It includes two basic components:
  - Interpretation of extracted patterns, possible visualization of the extracted patterns, removing redundant or irrelevant patterns, and translating the useful ones into terms understandable by users.
  - Consolidating and analysis discovered knowledge, incorporating this knowledge into the performance system, applying and deploying of the knowledge in the real scenario.

1.6 Literature Review

The researcher has made review of different related literatures i.e. books, journal articles, Conference proceeding papers, and the Internet in order to have detailed understanding on the present research trend in the study area.

Many IDS has been proposed by many researches using machine learning techniques to improve the IDS from different perspectives which is described in terms of detection of novel attack, detection accuracy, reduce false alarm rate, and time consuming. Authors in [14], has Proposed ensemble approach intrusion detection using Alternating Decision Tree (ADTree) and K-Nearest Neighbor (KNN) classifiers. Ensemble approach is a family of data mining that construct a set of classifiers and then classify new data points by taking a weighted vote of their predictions. It has proven very successful in designing high performance IDSs resulting on the mutual combination of multiple classifiers. Individual classifiers are built using both the input features space and additional features exploited from k-means clustering. The result shows that the method outperforms all the tests made especially with respect to R2L, U2R, and Probe classes.

Researchers in [15-18] have proposed hybrid prediction model by applying different classification and clustering algorithms. Their experiments were conducted using KDD cup99 and NSL-KDD dataset. In [15], Hybrid approach by combining different ML techniques has been implemented which aimed to increase attack detection rate while minimizing high false alarm rate. J48 (C45) with Random Tree, J48 (C45) with Random Forest, and Random Forest with Random Tree classifiers were used and the result have shown combining J48 with Random Tree improves performance of intrusion detection rate and false alarm rate. Author in [19] has conducted a comparative study on efficient intrusion detection system using hybrid approach which is by combining supervised and unsupervised DM techniques. Four data mining techniques, K-mean, fuzzy C mean,

naïve Bayes and support vector machine are applied. The researcher discovered Fuzzy C-Mean (FCM) is a better classification technique and SVM in terms of accuracy and detection rate. [16] has also proposed a hybrid machine learning technique for network intrusion detection based on combination of K-means clustering and support vector machine classification. The results have shown that the proposed technique has achieved a positive detection rate and reduce the false alarm rate. Hybrid intrusion detection method is proposed by [18] using C4.5 and support vector machine algorithms. The result demonstrate that the proposed method is better than the conventional methods in terms of the detection rate for both unknown and known attacks while it maintains a low false positive rate too.

To simulate an intrusion detection systems (IDS) model which meets the efficiency in real time traffic, reliable and enormous amount of dataset should be available. A dataset with a diverse and sizable amount of quality data which simulates the real time traffic can only help to train and test an intrusion detection system [20, 21]. [21] KDD was the pioneering benchmark dataset for evaluating a newly proposed intrusion detection system. It was created by the Defense Advanced Research Projects Agency (DARPA) and other interested institutions to provide the benchmark dataset for evaluation of a proposed intrusion detection system. This standard IDS dataset was used by the research community for many years as a benchmark for the evaluation of intrusion detection system. Author in [22] has been reviewed IDS researches made between 2010 and 2015 to investigate usage of this dataset in machine learning research (MLR). According to this review, most of the published studies have been used KDD99 and proved as the most used dataset in IDS and machine learning areas. In addition to KDD99, different datasets were also used by many researchers in this area of study. The most used datasets by researchers are listed in Table 1.2.

Table 1.2: Most used Datasets in IDS researches [22]

#	Dataset Name	Quantity of Papers
1	KDD99	133
2	NSL-KDD	23
3	DARPA	9
4	Kyoto	3

Another study was conducted by [7] to investigate statistical analysis on KDD99 dataset, they found two important deficiencies which highly affects the performance of evaluated IDS model, and resulted poor evaluation of anomaly detection approaches. It has huge number of redundant records which is about 78% and 75% of the records are duplicated in the train and test set, respectively. This will cause learning algorithms to be biased towards the more frequent records. The researcher employed seven ML models, each trained three times to analyze the records in KDD data set, 98% and 86% of the records in the train and test set were correctly classified respectively. This is due to random parts of the KDD train set are used as test sets which is the comparison of IDS models quite difficult. To solve the mentioned problems, NSL-KDD created which is the reduced version of version of KDD99 dataset.

The new version of KDD data set, NSL-KDD was created to eliminate the problems in KDD99 but still it has inherited some of the problem that is not perfect representative of existing real network. The KDD98 family evaluation dataset has been widely criticized by many researchers and they demanded new IDS dataset to overcome the existing deficiencies [7, 20-23].

A comprehensive evaluation of the existing datasets was conducted by [24] using evaluation criteria's and proposed an evaluation framework for IDS dataset. The defined evaluation framework is based on eleven features as shown in Figure 1.6. According to this evaluation framework, a complete dataset should cover eleven features or criteria

and the researchers concluded that a new dataset is needed as the existing datasets does not represent the existing real traffic, suffer from lack of traffic diversity and volumes, some of them do not cover the variety of attacks and they lack feature set and metadata. Figure 1.6 shows sample comparison between KDD and KYOTO datasets based on the evaluation criteria. Accordingly, both datasets have shown limitations.

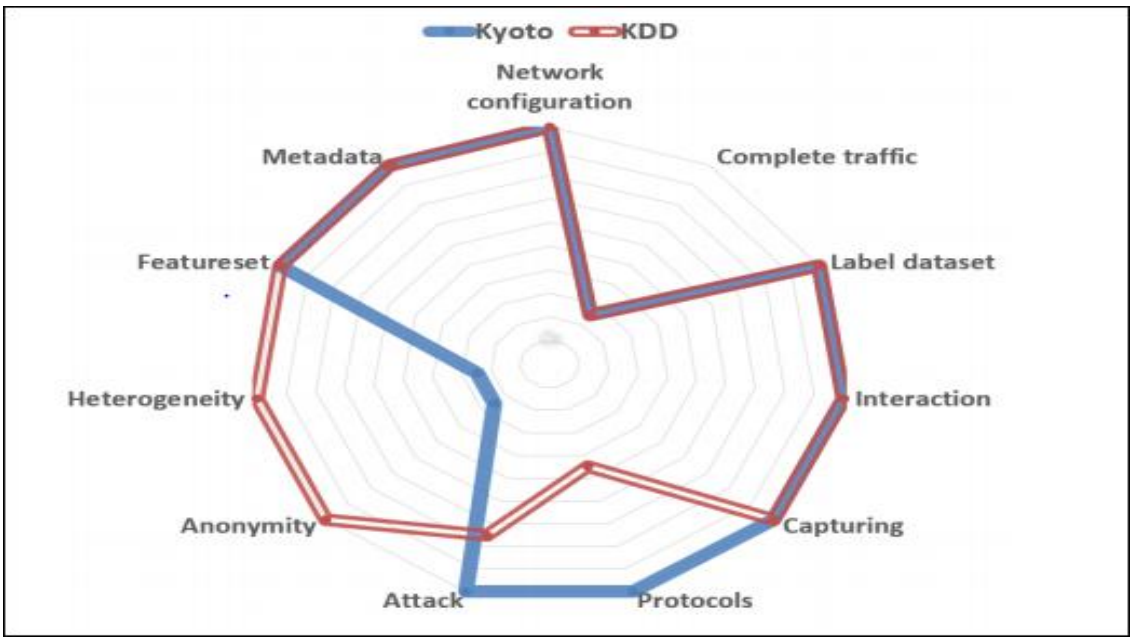


Figure 1.6: KDD99 and KYOTO datasets evaluation [24]

Based on the evaluation result which is conducted by [24], the state of the art IDS benchmark datasets KDD and others are no longer reliable, because they cannot meet the expectations of current advances in computer technology. The need to provide a reliable alternative IDS benchmark dataset motivated the University of New Brunswick institute of cyber security to generate an alternative dataset that matched modern computer technology and the corresponding developments in cyber-attack. As a result, a new CIC-IDS2017 benchmark dataset for the evaluation of machine learning based IDS models has been produced in 2017. It is prepared to solve the mentioned problems in the existing datasets [25]. It is prepared based on the framework proposed in [24].

From the literature review, we have observed that most of the IDS researches are used the KDD and improved version NSL-KDD for performance evaluation and abandoning this benchmark dataset and starting to use of newly introduced benchmark dataset for effective evaluation of machine learning based intrusion detection system is crucial.

This study will be conducted using the new CIC-IDS2017 benchmark dataset for the performance evaluation of IDS models.

1.7 Brief Thesis Outline

This thesis is structured into five chapters. Chapter one deals with introduction of the whole document. It states the statement of the problem, objective of the study, research methodology used and the literate review.

Chapter two describes about state of the art of ML algorithms and approaches and the third Chapter presents the system model used and dataset preparation, Data transformation and feature selection and evaluation metrics. The fourth Chapter presents the experiment result, performance analysis and evaluation of the models. The last chapter presents the conclusions and future work.

Chapter 2

Intrusion Detection System and Machine Learning Algorithms

Many types of IDS methods have been introduced for detection of network anomalies by monitoring and analyzing network traffic. Most of the methods are anomalies based which they identify deviations from an underlying normal traffic behavior [1]. According to the literature, the existing ML-based intrusion detection approach work basically in three basic approaches [2]:

- Unsupervised learning
- Supervised learning
- Hybrid learning

2.1 Unsupervised Learning

Unsupervised ML techniques facilitate the analysis of raw datasets, thereby helping in generating analytic insights from unlabeled data. It has many applications such as feature learning, data clustering, dimensionality reduction and outlier detection [26]. Clustering algorithms is one of unsupervised learning technique which significantly advance the state of the art in unsupervised ML techniques. In this study, Clustering algorithm is applied with the aim of anomaly detection or new attacks in computer networks [27].

Anomalous behavior is dynamic in nature. Novel types of anomalous behavior may emerge which needs labeled training data to be detected using supervised learning. However, clustering algorithm is method that is able to detect novel attack without any prior knowledge about it and is capable to find natural grouping of data based on similarities among the pattern [28]. Unlabeled dataset is taken as input for the cluster

algorithm and the method attempts to seek intrusion instances inside the data. Unsupervised learning is interesting technique as it uses unlabeled dataset where it can unconstraint us from the need of labeled data [26].

2.1.1 K-Means Algorithm

In data mining, clustering is the most important unsupervised learning technique used to find the patterns in a collection of unlabeled data. The k-Means clustering method is first used to partition the training instances into k clusters using Euclidean distance similarity. K-means clustering tends to distribute n observations into K clusters where each observation belongs to the nearest cluster. K is the number of clusters which is received as input from the user and n is the number of observations or values of the features in a dataset. The membership of an observation to a cluster is determined using the cluster mean. K-means clustering is used in numerous applications in the domains of network security and intrusion detection [1, 26]. On each cluster, representing a density region of normal or anomaly instances. The ML model is built using the clustered dataset as input where normal or attack label will be appended to the original dataset and this will help for the classifier to detect new anomalies [1]. The K-means background working process flow is shown in the below Figure 2.1.

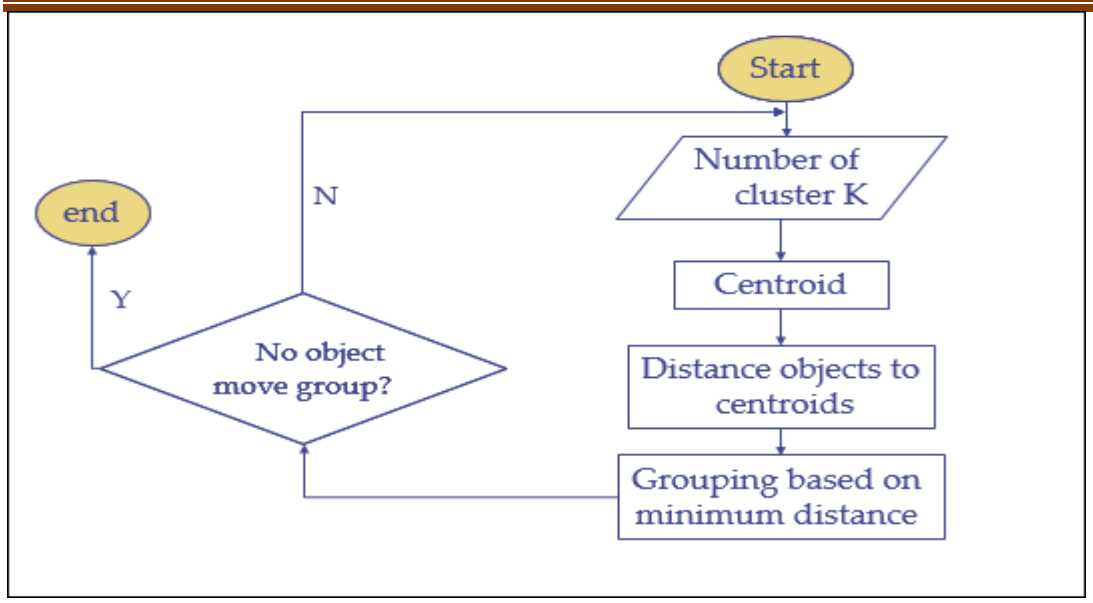


Figure 2.1: K-means process flow

2.2. Supervised Learning

In supervised learning, the dataset has instances labeled as normal or as belonging to anomaly classes. Supervised learning is an approach of ML which uses a set of predicting variables or features X as an input and a set of labels or predictive targets Y, where each entry in X must be labeled by an element of set Y. In this way, the ML models have a teacher in the form of set Y, which orients the model during the training process. New data instances are tested against this model to establish their membership in a class .This kind of ML is used frequently to solve classification and regression problems [1, 2]. Some of the most used supervised learning algorithms are decision tree (DT), neural network (NN) and support vector machine (SVM) [22]. In this study, DT and NN are selected as supervised learning techniques.

2.2.1 Decision Tree Algorithm

Tree based learning algorithms are considered to be one of the most used supervised learning methods in data mining tasks [29]. Decision trees are trees that classify instances

by sorting them based on feature weight and creates pattern of a data from the given training sample set. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a weight that the node can assume. Instances are classified starting at the root node and sorted based on their feature weight. It builds a tree using information theory, which choose splitting attributes from a dataset with the highest information gain. Some classical decision tree algorithms include C4.5, ID3, and CART. Decision tree techniques are composed of three basic components:

- A decision node specifying a test attributes.
- An edge or a branch corresponding to one of the possible attributes values.
- A Terminal Node, usually named an answer node, and it contains the class to which the object belongs.

To better illustrate the classification process, Figure 2.2 below illustrates simple example decision tree algorithm build the tree using the input features. The input features are features of the traffic data extracted from traffic captured data. These features are very important characteristics of the traffic data. Service, flag, source bytes, destination bytes and count are sample features taken from the CIC-IDS2017 dataset which is used in this study for performance evaluation of IDS models.

Input features

Service	Flag	Src_bytes	Dst_bytes	Count	Srv_count	Label
18	10	491	0	2	2	normal
43	10	146	0	13	1	normal
48	6	0	0	123	6	Attack
22	10	232	8153	5	5	normal
22	10	199	420	30	32	normal
48	2	0	0	121	19	Attack
48	6	0	0	166	9	Attack
48	6	0	0	117	16	Attack
50	6	0	0	270	23	Attack

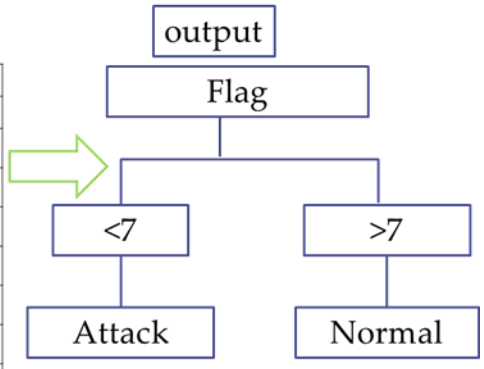


Figure 2.2: Decision Tree training input and output Example

Decision trees use multiple algorithms to decide to split a node in to two or more sub nodes. The most commonly used algorithms in decision tree are:

- **Gini Index:** The Gini index is commonly used to measure the discriminative power of a particular feature. The Gini index (S) for a set S of data points may be computed on the class distribution $p1 \ldots p_k$ of the training data points in the node. Let p_j be the fraction of records in S of class c

$$p_j = \frac{|\{t \in S:\ t[C] = c_j\}|}{|S|}$$

Then

$$G(S) = 1 - \sum_{j=1}^{k} p^2{}_j \quad \ldots \quad (1)$$

Where k is the number of classes in the dataset

- **Chi-Square:** It is an algorithm to find out the statistical significance differences between sub-nodes and parent node. It is measured by sum of squares of standardized differences between observed and expected frequencies of target variable.

$$x^2 = \sum_{j=1}^{k} \frac{(O_i - E_i)^2}{E_i} \quad \ldots \quad (2)$$

Where O is the number of observations in a class and E is the number of expected observations in a class and i is the "ith" position in the dataset record.

- **Entropy:** The class-based entropy measure is related to notions of information gain resulting from fixing a specific attribute value. The entropy measure achieves a similar goal as the Gini index at an intuitive level, but it is based on sound information-theoretic principles. Measures the expected reduction in entropy of class before and after observing features. Larger difference indicates that the

selected feature is more important to contain the class discriminatory information[29].

$$E = \sum_{j=1}^{k} p_j \log_2(p_j) \quad \text{(3)}$$

- **Reduction in Variance:** Reduction in variance is an algorithm used for continuous target variables (regression problems). This algorithm uses the standard formula of variance to choose the best split. The split with lower variance is selected as the criteria to split the population.

2.2.2 Neural Network Algorithm

Neural Networks (NN) are important data mining tool used for classification and clustering. It is an attempt to build machine that will mimic brain activities and be able to learn. NN usually learns by examples. If NN is supplied with enough examples, it should be able to perform classification and even discover new trends or patterns in data. Basic NN is composed of three layers, input, output and hidden layer. Each layer can have number of nodes and nodes from input layer are connected to the nodes from hidden layer. Nodes from hidden layer are connected to the nodes from output layer. Those connections represent weights between nodes [29].

The back-propagation algorithm is an extension of the least mean square (LMS) algorithm that can be used to train multi-layer networks. Both LMS and back-propagation are approximate steepest descent algorithms that minimize squared error. The only difference between them is in the way in which the gradient is calculated. The back-propagation algorithm uses the chain rule in order to compute the derivatives of the squared error with respect to the weights and biases in the hidden layers. It is called back-propagation because the derivatives are computed first at the last layer of the network, and then propagated backward through the network, using the chain rule, to compute the derivatives in the hidden layers. The logic behind this algorithm is just the output of

NN is evaluated against desired output. If results are not satisfactory, connection (weights) between layers are modified and process is repeated again and again until error is small enough [29, 30].

Figure 2.3 represent architecture of a simple NN. It is made up from four inputs, five hidden layers and output.

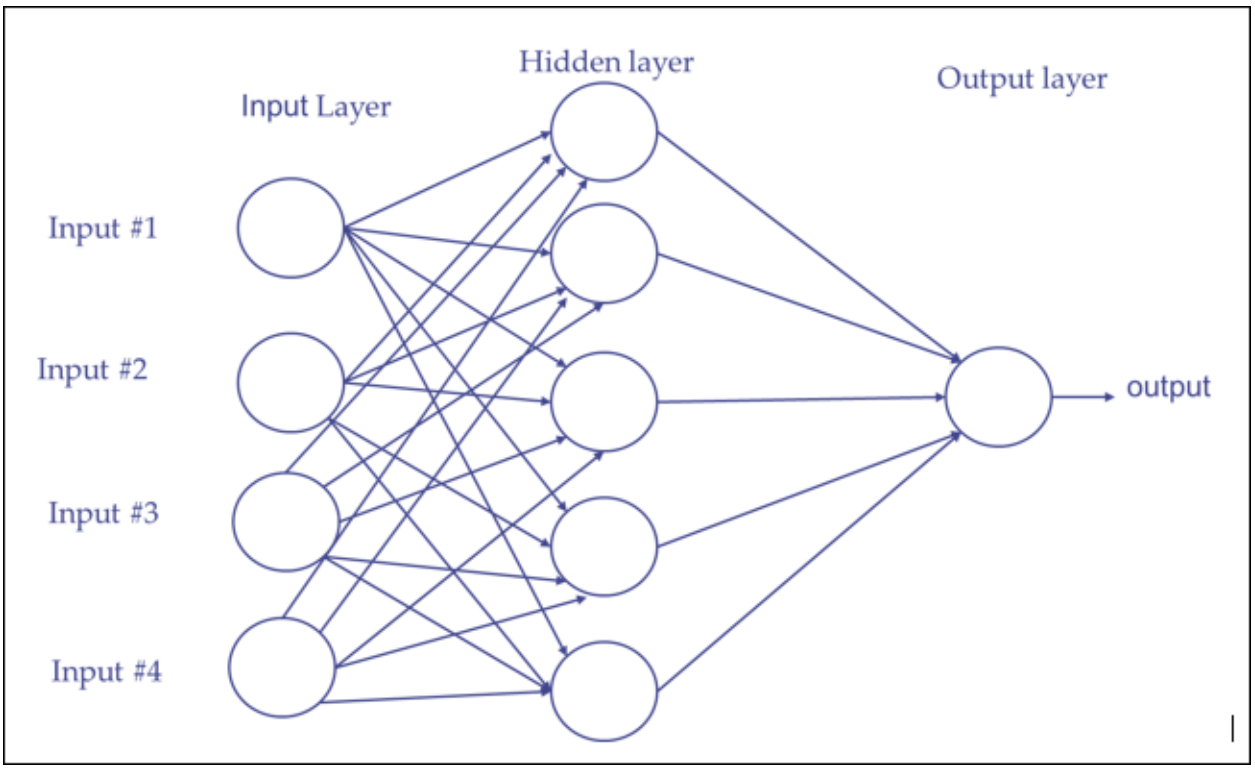


Figure 2.3: Basic neural network architecture [26]

2.3 Hybrid Learning

A hybrid intrusion detection method combines supervised and unsupervised network intrusion detection methods. Supervised methods have the advantage of being able to detect known attacks with high accuracy and low false alarms. On the other hand, an unsupervised method has the ability to detect novel attacks. Thus, a hybrid intrusion detection method is capable of detecting both known as well as unknown attacks with high accuracy [1]. A hybrid NIDS attempts to overcome the shortcomings of super-vised

and unsupervised NIDSs by taking advantage of the benefits of both approaches. This approach is applied in this study as it is new techniques and many studies are performed using the first and the second approaches. So, the ML models are built based on the following three hybrid approaches:

- k-Means clustering followed by Decision Tree classification
- k-Means clustering followed by Neural Network classification

Chapter 3

System Model and Implementation

3.1 System Model

The application of different ML models to solve a problem and making decisions data driven based is become the most important agenda. The most commonly used approaches to develop ML model are classified in three categories. The first one is using only one ML algorithm, that is either supervised or unsupervised learning. The second approach is hybrid which uses both a supervised learning and unsupervised learning algorithm. The aim of this approach is to have both algorithms complement each other in order to improve the performance of the model with respect to a certain task. The third approach is called ensemble learning which is when multiple ML algorithms are used to build ML model, these being either supervised or unsupervised learning algorithms

In this study, hybrid ML model is proposed which is a combination unsupervised and supervised algorithm. It is also applied feature selection techniques to enhance the performance of the ML models. The aim of this approach is to construct hybrid ML model which is to maximize intrusion detection in computer networks, where known attacks are detected by supervised learning and unknown attacks are detected by unsupervised learning technique. The K-means algorithm will cluster the input features in to normal and attack. The clustered input feature will be used by the classifier algorithm. The clustered value will be considered as an input feature by the classifier algorithm and thus will able the classifier to detect new anomalies. Thus, both algorithms will complement each other in order to improve the performance of the model in detecting intrusions in computer network. This research was conducted based on the system model shown in Figure 3.1. The ML models are implemented according this system models using the

Python programming language which is the whole process of changing low level data into high level knowledge. The model construction is based on an iterative and experimental that allows to choose the models that adjust the best to the scenario.

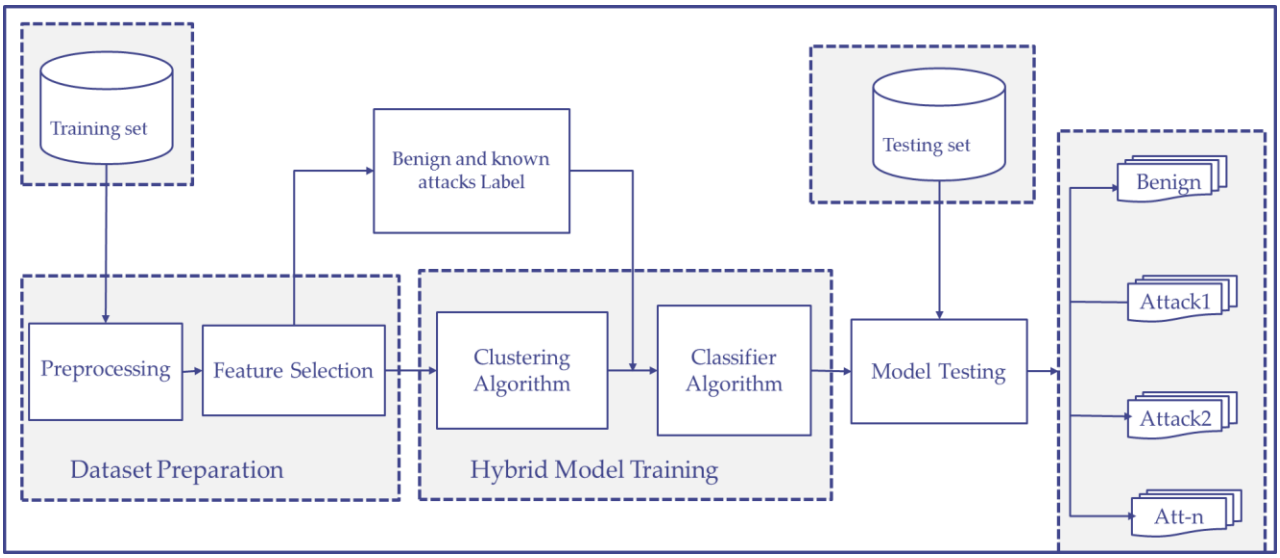


Figure 3.1: System Model [1]

The initial step in the KDD process flow is the data selection which is the critical step in data mining as the output we get highly depends on the data quality we input to the model. Once the data is selected, it is sent for preprocessing in another module. In the Preprocessing step, data cleaning, imputation and feature scale or normalization are performed. Feature selection techniques are applied to identify contributing features for the model. After the important features are selected using the feature selection techniques, the unsupervised learning is applied and the clustered data is sent to the supervised learning. In general, the training work flow used in this study, shown in Figure 3.2 is composed of five steps: the first step is dataset selection, where we used the CIC-IDS2017 dataset. The second step is Preprocessing, where data cleaning and feature scaling is performed. The third step is feature selection where important features of input dataset is selected using either PCA or FI. Fourth step is clustering, where the dataset is

clustered in to two centroids: normal and attack and these labels are appended in to the dataset to be used by the classifier in the training. The last step in the training phase is training the classifier, the clustered dataset will be trained by the classifier which the clustering label will be used as a feature input and used to enable the classifier detect the new anomalies.

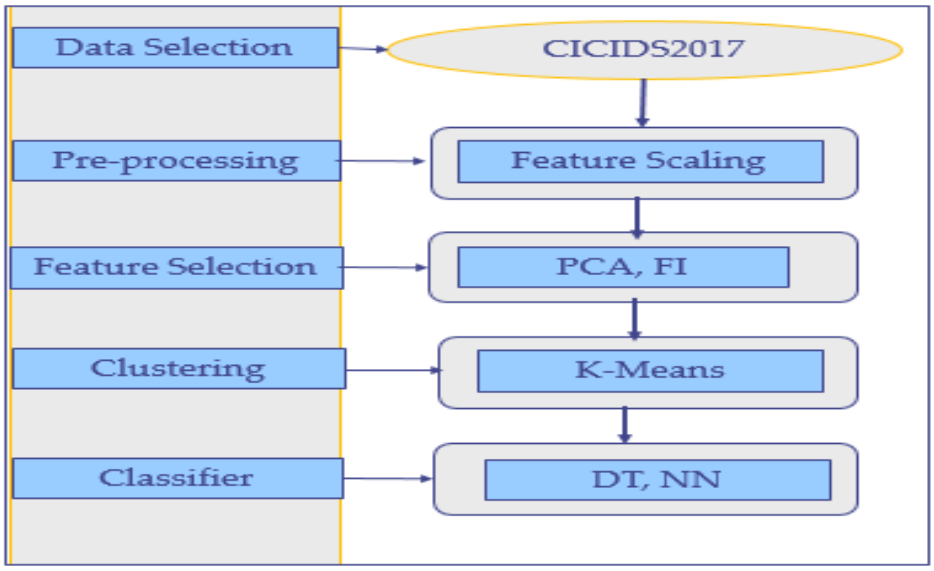


Figure 3.2 Process flow used for training model [2]

The second part of the ML model construction, the testing workflow used in this study shown in Figure 3.3 which consists of six steps. The first to three steps will be applied to the test dataset which is split CIC-IDS2017 dataset the same to the training phase applied. The fourth step in this phase is the validation which is performed using K-fold cross validation. The fifth step is the prediction where the training model will be tested using the test data. The last step is the model evaluation, where the predictions made during step 5 are evaluated using the performance metrics (confusion matrix).

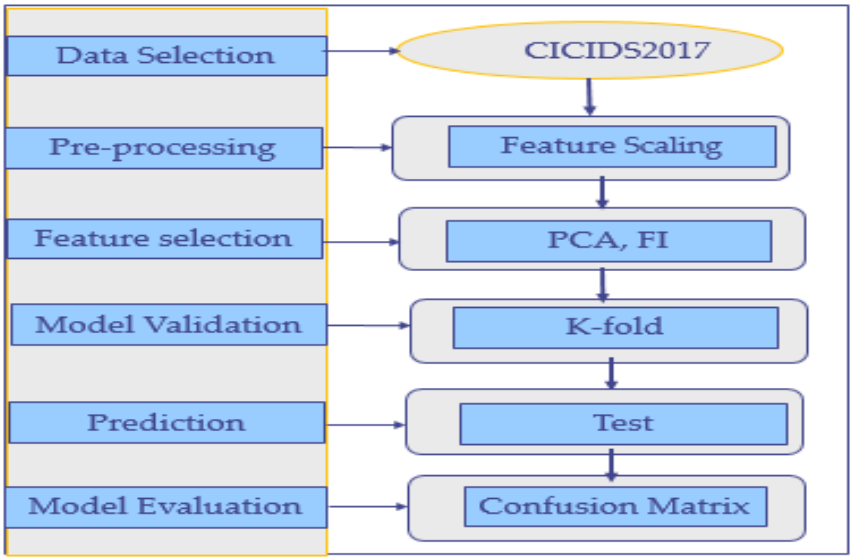


Figure 3.3: Process flow used for testing model [2]

3.2 Dataset Description

As stated in chapter one, under the research methodology section of this thesis, the ML process model applied in this study is the KDD model which starts from selection of data. The dataset applied in this research was collected from Canadian Institute for Cyber security (CIC) based at the University of New Brunswick. The CIC-IDS2017 dataset is produced to overcome the shortcomings discussed in the literature review. It was aimed to generate diverse and comprehensive benchmark dataset for intrusion detection based on the creation of user profiles which contain abstract representations of events and behaviors seen on the network [31].

The data capturing period started at 9 a.m., Monday, July 3, 2017 and ended at 5 p.m. on Friday July 7, 2017, for a total of 5 days. Monday is the normal day and only includes the benign traffic. The implemented attacks include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. A total of 2843553 instances were collected and are currently stored in Comma Separated Values (CSV) files, one per day. It is prepared from packet capture (PCAP) files using Cyclometer, a network traffic flow

generator which has been developed in the University of New Brunswick. This dataset contains the 2843553 instances with 85 attributes or features captured over the five days. Each instance holds the information of an IP flow generated by a network device i.e. Statistical flow features (such as Duration, Number of packets, Number of bytes, and Length of packets, among others). Most of the attributes are numeric type but there are also nominal types and a date type due to the Timestamp. The dataset presented in this study is available at [31].

As illustrated on Table 3.1, the dataset attributes are grouped in to seven which proves that the CIC-IDS2017 dataset is more complete than the KDD99 [32]. It is a reliable dataset that contains benign and recent common attack network flows, which meets real world criteria. It solves the limitations like some of the existing datasets suffer from lack of traffic diversity and volumes, some of them do not cover the variety of attacks, while others anonymized packet information and payload which cannot reflect the current trends, or they lack feature set and metadata [25].This new dataset is publicly available for computer network security and related research works [31].

Table 3.1: Dataset attributes and description [31]

Group of attributes	Attributes	Description
Network identifiers (7 attributes)	FlowID; Source IP; Source Port; Destination IP; Destination Port; Protocol; Timestamp	These attributes hold all the information related to the source and destination of an Internetflow, i.e., IP addresses, transport layer protocol and ports
Flow descriptors (34 attributes)	Total Fwd Packets; Total Bwd Packets; Total Length of Fwd Packets; Total Length of Bwd Packets; Fwd Packet Length Max;	These attributes hold all information related to the

Group of attributes	Attributes	Description
	Fwd Packet Length Max; Fwd Packet Length Min; Fwd Packet Length Mean; Fwd Packet Length Std; Bwd Packet Length Max; Bwd Packet Length Min; Bwd Packet Length Mean; Bwd Packet Length Std; Flow Bytes S; Flow Packets S; Min Packet Length; Max Packet Length; Packet Length Mean; Packet Length Std; Packet Length Variance; Down Up Ratio; Avg Fwd Segment Size; Avg Bwd Segment Size; Fwd Avg Bytes Bulk; Fwd Avg Packets Bulk; Fwd Avg Bulk Rate; Bwd Avg Bytes Bulk; Bwd Avg Packets Bulk; Bwd Avg Bulk Rate; Init Win bytes forward; Init Win bytes backward; act data pkt fwd; min seg size forward; Label	internet flow, i.e., number of packets, volume and standard deviation among others in the forward and backward direction.
Inter-arrival times (15 attributes)	Flow Duration; Flow IAT Mean; Flow IAT std; Flow IAT Max; Flow IAT Min; Fwd IAT Total; Fwd IAT Mean; Fwd IAT Std; Fwd IAT Max; Fwd IAT Min; Bwd IAT Total; Bwd IAT Mean; Bwd IAT Std; Bwd IAT Max; Bwd IAT Min	These attributes hold all the information related to the interarrival times in the forward and backward direction
Flag features (12 attributes)	Fwd PSHflags; Bwd PSHflags; Fwd URGflags; Bwd URGflags; FIN Flag Count; SYN Flag Count; RST Flag Count; PSH Flag Count; ACK Flag Count; URG Flag Count; CWE Flag Count; ECE Flag Count	These attributes show the information related to all the flags contained in the header of the packets, i.e., Pushflags, Urgent flags, Finishflags, among others
Sub-flow descriptors (4 attributes)	Subflow Fwd Packets; Subflow Fwd Bytes; Subflow Bwd Packets; Subflow Bwd Bytes	If there were subflows, these attributes present all

Group of attributes	Attributes	Description
		the information related to their number of packets per flow and volume in the forward and backward direction
Header descriptors (5 attributes)	Fwd Header Length; Bwd Header Length; Average Packet Size; Fwd Header Length 1	Among these attributes the information related to the header is stored
Flow timers (8 attributes)	Active Mean; Active Std; Active max; Active min; Idle Mean; Idle std; Idle max; Idle min	These attributes store the information related with the time each flow was active and inactive

3.2.1 Data Selection

CIC-IDS2017 dataset contains benign and the most up-to-date common attacks, which resembles the true real-world data (PCAPs). It also includes the results of the network traffic analysis using CICFlow Meter with labeled flows based on the time stamp, source and destination IPs, source and destination ports, protocols and common attack types [31]. The CIC-IDS2017 dataset consists both quantitative and qualitative 85 features which are different from the previous versions of benchmark datasets. All records are labeled as either normal or an attack. The total dataset record label falls in to one of the following five main classes for intrusion analysis, one benign class and four main intrusion classes:

I. **BENIGN**: is realistic background traffic generated B-Profile system which is proposed by [33] . The B-Profile is responsible for profiling the abstract behavior of human interactions and generate a naturalistic benign background traffic. It is generated by applying to a diverse range of network protocols with different topology to represent the abstract properties of human and attack behavior.

II. **Denial of service (DoS)**: is type of attack meant to shut down a machine or network, making it inaccessible to its intended users which is accomplished by flooding the targeted machine in an attempt to overload it. DoS attacks do not typically result in the theft or loss of significant information or other assets, cause the victim a great deal of time and money to handle. Examples are Apache2, Mail bomb, SYN Flood, ICMP flood and Ping of death.

III. **Distributed Denial of service (DDoS):** DDoS allows for exponentially more requests to be sent to the target, therefore increasing the attack power. It also increases the difficulty of attribution, as the true source of the attack is harder to identify.

IV. **Brute Force Attack:** This is one of the most popular attacks that only cannot be used for password cracking, but also to discover hidden pages and content in a web application. It is basically a hit and try attack using software tools, then the victim succeeds. It is becoming common attack in webs using the SQL Injection, which an attacker can create a string of SQL commands, and then use it to force the database to reply the information.

V. **Botnet:** A number of Internet connected devices used by a botnet owner to perform various tasks. It can be used to steal data, send spam, and allow the attacker access to the device and its connection.

The total dataset distribution of CIC-IDS2017 is presented as shown in Table 3.2.

Table 3.2: Distribution of CIC-IDS2017 dataset

#	Dataset Label	Number of Records	Remark
1	BENIGN	2372132	
2	DDoS	200767	
3	DoS	252673	
4	Botnet	1966	
5	Brute force	16015	
6	**Total record**	**2843553**	

The number of datasets selected for this research purpose is around 10% of the total CIC-IDS2017 dataset record. The selected dataset is summarized in Table 3.3.

Table 3.3: Distribution of selected dataset from CIC-IDS2017

#	Dataset Label	Number of Records selected	Remark
1	BENIGN	98746	
2	DDoS	94213	
3	DoS	63268	
4	Botnet	1966	
5	Brute force	16015	
6	**Total record**	**274208**	

3.3 Data Cleansing and Preprocessing

The quality of the data and the amount of useful information that it contains are key factors that determine how well a machine learning algorithm can learn. Therefore, the selected dataset should be clean and preprocessed before it is feed to a machine learning algorithm for training [29, 34]. Preprocessing will help to prevent differing magnitude among variables from producing erroneous statistics. During the data preprocessing, it is started with the data exploration, checked about missing data and imputed missing values in all columns, feature scaled in to the same scale and transformed in to new

features. There are many essential data preprocessing techniques that help us to build good machine learning models:

- Feature scaling
- Standardization
- Imputation of missing values

Feature scaling is a crucial step in preprocessing pipeline which bring features onto the same scale. Normalization is the process of scaling individual samples to have unit norm which transform attributes with a Gaussian distribution and differing means and standard deviations to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1 [35]. In this study, the predicting variables were normalized with feature scaling typically done via Equation (4) so that variables are scaled between 0 and 1, where X represents the current value of the entry, Xmax and Xmin are the maximum and minimum value of the entry and Xsc is the scaled value of the entry.

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (4)$$

The other important technique of data pre-processing is handling missing values in a dataset. Most computational tools are unable to handle such missing values or would produce unpredictable results if we simply ignored them. Therefore, it is crucial that we take care of those missing values before we proceed with further analyses. There are several techniques for dealing with missing values; removal of samples or dropping of entire feature columns is simply not feasible, because we might lose too much valuable data. Imputation of missing values in dataset is recommend technique. In this paper imputation technique is applied to fill missing values in the selected dataset for model evaluation. It uses interpolation technique to estimate the missing values from the other samples in our dataset. One of the most common interpolation techniques is mean

imputation, where we simply replace the missing value by the mean value of the entire feature column [35].

As a result of preprocessing, the data were brought to a form convenient for further work with machine learning methods. The main statistical characteristics of the numerical data (the number of unallocated values, mean, standard deviation, range, median, 0.25 and 0.75 quartiles) are shown in Figure 3.4. Analyzing these data, we can conclude that we have a complete set of data (the number of records is the same for each column or attributes of the dataset, which indicates the absence of omissions in the data, their completeness).

	Source Port	Destination Port	Protocol	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packe Length Mi
count	590286.000000	590286.000000	590286.000000	5.902860e+05	590286.000000	590286.000000	590286.000000	5.902860e+05	590286.000000	590286.00000
mean	42683.544719	7048.820509	9.720269	1.567735e+07	10.007712	11.315225	453.849354	1.801146e+04	162.406552	15.82078
std	21715.592936	17547.898105	5.209405	3.441685e+07	753.622349	1018.789164	5080.980076	2.282183e+06	439.293015	34.87696
min	0.000000	0.000000	0.000000	-4.000000e+00	1.000000	0.000000	0.000000	0.000000e+00	0.000000	0.00000
25%	35938.000000	53.000000	6.000000	1.890000e+02	2.000000	1.000000	2.000000	0.000000e+00	2.000000	0.00000
50%	51789.000000	80.000000	6.000000	4.737000e+04	2.000000	2.000000	62.000000	1.410000e+02	37.000000	0.00000
75%	58498.000000	443.000000	17.000000	5.027245e+06	5.000000	5.000000	286.000000	1.958000e+03	181.000000	36.00000
max	65535.000000	65505.000000	17.000000	1.200000e+08	206446.000000	276072.000000	2428415.000000	6.270000e+08	24820.000000	2065.00000

8 rows × 78 columns

Figure 3.4 Main statistical characteristics

The matrix of correlation of numerical features is the form of the data matrix, which includes correlation coefficients for all pairs of analyzed variables. The correlation matrix is the basis for factor analysis, canonical correlation, and other statistical techniques that reproduce the structure of the relationship between variables or input features. A visual

display of the correlation matrix of the selected CIC-IDS2017 dataset is given in Fig. 3.5.

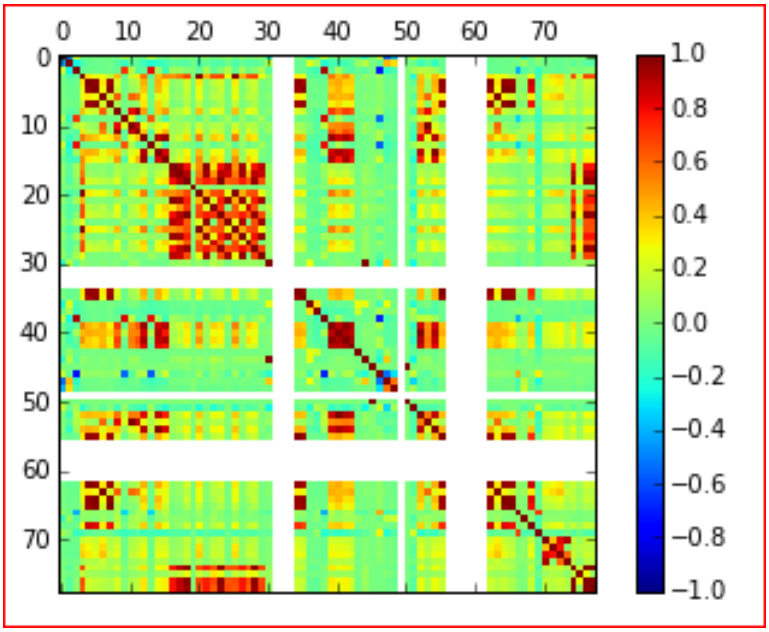


Figure 3.5: Matrix of correlation between features

3.4 Feature Selection and Transformation

The quantification of the feature importance (FI) is a crucial issue not only for ranking the features before a stepwise estimation model but also to interpret data and understand underlying phenomenon in many applied problems. Feature selection is an integral component of knowledge discovery and machine learning which allow to identify features of a dataset that truly provide information to the ML model; reducing the dimensionality of the information with the objective of obtaining faster and more accurate models. Reducing the dimensionality of the data, reduces the size of the hypothesis space and thus results in faster execution time. It is a multistep process particularly useful in analyzing real life high-dimensional data such as network intrusion data, biological data and criminal investigation data. Applying effective feature selection is so important step in KDD model for intrusion detection datasets for constructing high performance IDS model. The performance of IDS would be improved by applying feature

selection methods [1, 2, 30]. In this study, we have applied two common feature selection techniques:

- Principal component analysis(PCA)
- Feature importance (FI)

3.4.1 Principal Component Analysis (PCA)

PCA is an unsupervised linear transformation technique that is widely used across different fields, most prominently for dimensionality reduction. PCA is a statistical procedure that orthogonally transforms the original n coordinates of a data set into a new set of n coordinates called principal components. It finds a low-dimensional representation of a dataset based on highest variation by use of a covariance matrix. The most important features are represented as the first columns from left to right in the covariance matrix. Keeping only the first $m < n$ components reduces the data dimensionality while retaining most of the data information based on the variation in the data [2, 36].

3.4.2 Feature Importance (FI)

Random forest is well established classification model that provide rich feature selection techniques called feature importance in addition to predictive function. One approach to dimensionality reduction is to generate a large and carefully constructed set of trees against a target attribute and then use each attribute's usage statistics to find the most informative subset of features. It uses the mean decrease impurity or mean decrease accuracy metric to construct a feature importance measure, apparently to measure the prediction strength of each feature [36]. In this method the weight of each feature is calculated by using a feature importance method. Then the features with weights larger than a predefined threshold are selected. The selected relevant features are then subject

to the second step which is training the ML model with the relevant features. The following are the steps used to extract important features using random forest classifier:

- Load the processed dataset.
- Split the data into training and test sets.
- Create a random forest classifier and train the classifier.
- Print the feature name and gini importance of each feature.
- Identify and select most important features using a selector object that will use the random forest classifier to identify and select a threshold value that includes the important features.
- Train the selector after the selector object and threshold value is defined.
- Create a data subset with only the most important features- transform the data to create a new dataset containing only the most important features to both the training and test data.
- Train and test the desired ML algorithms using the transformed new dataset with most important features.

Sample features name and corresponding information weight is presented in Figure 3.6.

```
[(0.13572978147093223, 'Active Min'),
 (0.11590532722897642, 'Active Mean'),
 (0.10244101034762014, 'Idle Max'),
 (0.06571663525744227, 'Idle Min'),
 (0.05500584641466264, 'Fwd IAT Total'),
 (0.05071391006168792, 'Fwd Packets/s'),
 (0.04797671955217827, 'Idle Mean'),
 (0.04082622449381943, 'Flow IAT Max'),
 (0.04019617145015077, 'Fwd IAT Max'),
 (0.028796186189355578, 'Flow Duration'),
 (0.019690038194689813, 'Source Port'),
 (0.018001860980101526, 'Packet Length Mean'),
 (0.016701033101200642, 'Fwd IAT Min'),
 (0.01664525283488781, 'Active Max'),
 (0.016000028895304743, 'Bwd Packet Length Max'),
 (0.013826037753567475, 'Fwd IAT Mean'),
 (0.012286595550073618, 'Flow IAT Mean'),
 (0.012113412483826507, 'Bwd IAT Max'),
 (0.011443373199807608, 'Fwd IAT Std'),
```

Figure 3.6: Sorted features names with respect to informative weight

It was found that all features in the CIC-IDS2017 dataset does not contribute for intrusion detection learning model and 12 features have 0 weight. Using these features as input to the ML model has no impact in the accuracy of the model but will consume computing resource. Features with 0 information weight are illustrated in figure 3.7.

```
(0.0, 'RST Flag Count'),
(0.0, 'Fwd URG Flags'),
(0.0, 'Fwd Avg Packets/Bulk'),
(0.0, 'Fwd Avg Bytes/Bulk'),
(0.0, 'Fwd Avg Bulk Rate'),
(0.0, 'ECE Flag Count'),
(0.0, 'CWE Flag Count'),
(0.0, 'Bwd URG Flags'),
(0.0, 'Bwd PSH Flags'),
(0.0, 'Bwd Avg Packets/Bulk'),
(0.0, 'Bwd Avg Bytes/Bulk'),
(0.0, 'Bwd Avg Bulk Rate')]
```

Figure 3.7: Features names with 0 informative weight

3.5 Performance Evaluation Metrics

After doing the data selection, bringing feature to the same scale and feature selection techniques is applied, Selection of ML algorithms, and of course, implementing a model and getting some output in the forms of a class, the next step is to find out how effective is the model based on some metrics using test datasets. The metrics we choose to evaluate ML model is very important as it influences how the performance of the model is measured and compared. The goal of intrusion detection model is to minimize false alarms rate or to maximize detection rate. [37] high detection rate and low false positive rate are the key criteria of any ML based intrusion detection model.

3.5.1 Confusion Matrix

A confusion matrix summarizes the number of instances predicted correctly or incorrectly by a classification model. We used to evaluate the intrusion detection model, the standard metrics derived from the confusion matrix table are; true positive (TP), true negative (TN), false positive (FP) and false negative (FN). In this study, there are five classes (i.e. Benign, Botnet, DDoS, DoS and Brute force) and therefore the matrix has dimensions of 5×5. A confusion matrix is similarly defined in that row and column 5×5 matrix for the CIC-IDS2017 dataset. Table 3.4 shows the confusion matrix for intrusion detection.

Table 3.4: Confusion Matrix for intrusion detection[38]

Confusion Metrics		**Predicted**	
		Normal	Intrusions (attacks)
Actual	Normal	True Negative (TN)	False Alarm (FP)
	Intrusions (attacks)	False Negative (FN)	Correctly detected Attacks (TP)

The numbers of true positive (TP), false negative (FN), false positive (FP), and true negative (TN) for each class will be calculated based on below the principles:

- The total number of test samples of any class would be the sum of the corresponding row (i.e. the TP+FN for that class).
- The total number of FN's for a class is the sum of values in the corresponding row excluding the TP.
- The total number of FP's for a class is the sum of values in the corresponding column except TP
- The total number of TN's for a certain class will be the sum of all columns and rows excluding that class's column and row.

Terminologies associated with Confusion matrix are:

- **True Positives (TP) -** True positives are the cases when the actual class of the data point was 1 (True) and the predicted is also 1 (True). From the context of this study, it defines number of malicious records that are correctly identified.
- **True Negatives (TN) -** True negatives are the cases when the actual class of the data point was 0 (False) and the predicted is also 0 (False). From the context of this study, it defines the number of benign records that correctly classified.
- **False Positives (FP) -** False positives are the cases when the actual class of the data point was 0 (False) and the predicted is 1 (True). False is because the model has predicted incorrectly and positive because the class predicted was a positive one. In the context of this study, it defines the number of records that are identified as attacks but in fact they are benign activities.
- **False Negatives (FN) -** False negatives are the cases when the actual class of the instance was 1 (True) and the predicted is 0 (False). False is because the model has predicted incorrectly and negative because the class predicted was a negative one

(0). In the context of this study, it defines the number of records that are incorrectly classified as legitimate activities however in fact they are malicious.

3.5.2 Metrics Derived from Confusion Matrix

Below are computation metrics of classification model which are derived from the confusion matrix in Table 3.4.

3.5.2.1 Accuracy

Accuracy is used to evaluate the performance of an IDS in terms of correctness. It measures the ability of a classifier in correctly identifying all samples, no matter it is positive or negative. It determines the proportion of correctly classified instances in relation to the total number of instances of the test. From the confusion matrix in Table 3.4, we can say that accuracy is the percentage of correctly classified instances over the total number of instances in total test dataset[1, 2].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} * 100 \quad \text{..... (5)}$$

Due to the nature of attacks divers and dynamic change and being the number of attack traffic instances is smaller than normal traffic instances, the IDS system generates a large amount of false alarms. Measuring a performance of classifier ML model using accuracy metrics would not be enough. Thus, a ML model should be mastered from different aspects to indicate its performance.

3.5.2.2 Detection Rate (DR)

Detection rate refers to the proportion of attack detected among all attack sample instances. Detection rate may be termed as the true positive rate (TPR), Sensitivity or recall. It is the frequency of correctly classified positive samples among all real positive samples. It measures the ability of a classifier in identifying positive samples. Thus, the detection accuracy is defined as follows [39]:

$$DR = \frac{TP}{TP+FN} * 100 \ldots\ldots (6)$$

3.5.2.3 Specificity

Measures the ability of a classifier in identifying negative samples. It defines the proportion of test data that had malicious, were predicted by the models as non-malicious. Specificity may be termed as the true negative rate (TNR) [2, 39].

$$Specificity= \frac{TN}{TN+FP}*100 \ldots\ldots (7)$$

3.5.2.4 Error Rate

The error rate measures the percentage of incorrectly classified entries.

$$Error\ rate= \frac{total\ test\ data - Total\ correctly\ classified}{total\ test\ data}*100 \ldots\ldots (8)$$

3.5.2.5 False Positive Rate (FPR)

False positive rate, also termed as False Alarm Rate (FAR) which measures normal attack samples identified as an attack or it is the proportion that normal data that is falsely detected as attack to the total samples(attack and normal instances) [30].

$$FPR= \frac{FP}{FP+TN}*100 \ldots\ldots (10)$$

3.5.2.6 Precision

Precision indicates the frequency of true positives among all positive outputs. It shows the number of class members classified correctly over the total number of instances classified as class members or shows what proportion of the test data having malicious, actually had malicious. In the context of intrusion detection, precision measures the

effectiveness of IDS in identifying anomalous and normal instances in a test dataset. Precision also termed as positive predictive value (PPV) [1, 30].

$$\text{Precision}=\frac{TP}{TP+FP}*100 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (11)$$

Chapter 4

Model Experimentation

This chapter describes the training of the models and experiments made based on the described procedure in the previous chapters which followed the KDD model. According to KDnuggets blog [40], a Poll had made to assess the current leading software for data science and machine learning in 2018. Accordingly, python programing language got 65% share of the available ML tools (Weka, orange, Java and R). This shows, it is the most popular machine learning tool currently. The reason behind the popularity of this programing language is due to the most enterprise solutions are powered using Python programing language and ML model built with this tool will be easily interoperable with the existing enterprise solutions. I used the Python programming language because of the great amount of packages for ML tasks with ample libraries and documentation is easily available. We have used a standard dataset, CIC-IDS2017 intrusion detection system dataset which is prepared by Canadian Institute for Cyber security [31]. It is a new dataset which is prepared to solve the limitations of KDD98 based datasets. CIC-IDS2017 dataset is around 2.8 million records and we used around 10% of it. The dataset is selected to maintain the balance of dataset record between classes. Input dataset is split for training and testing phases as shown in Table 4.1 after the necessary preprocessing and feature selection techniques are applied.

Table 4.1: Input dataset split

Input Features (X)	Target Class (Y)	% Split
X-training	Y-training	70%
X-test	Y-test	30%

4.1 Experiment Setup

All experiments were performed in a computer with the configurations Intel(R) Core (TM) 2 CPU 2.16GHz, 16 GB RAM, and the operating system platform is Microsoft server 2012. Classification models are implemented using python common machine learning libraries (i.e. NumPy, SciKit-Learn, Pandas and Matplotlib) that contain libraries for data preprocessing, classification, clustering, and visualization. In this study three types of scenarios have been used to build the hybrid ML models as shown in Table 4.2.

Table 4.2: Experiments scenarios used to build hybrid ML model

	Feature selection Techniques	Unsupervised ML	Supervised ML
Scenario I	PCA	K-Means	DT
	PCA	K-Means	NN
Scenario II	FI	K-Means	DT
	FI	K-Means	NN
Scenario III	-	K-Means	DT
	-	K-Means	NN

4.2 Models Experimentation

In a data mining classification problem, the predictive model will classify instance of a dataset into a particular class. An IDS model attempts to classify all traffic as either normal or attacks. The challenges in IDS model is to be able to minimize the number of false positives (classification of normal traffics as attack) and also false negatives (classification of attack traffics as normal).

This research paper proposes the design and implementation of four hybrid ML models that use the unsupervised algorithm (K-Means) in the first level and supervised learning algorithms (DT and NN) in the second level. The PCA and FI algorithms are feature selection algorithms we use. We choose the hybrid approach with the goal of the

intrusion detection maximization in computer networks, where known attacks are detected by supervised learning and unknown attacks are detected by unsupervised learning. The proposed hybrid model combines unsupervised algorithm K-means with supervised algorithms shown in Table 10. At the first level the model classifies the entries using K-Means first in order to form groups with two labels (i.e. Normal and attack). At this classification level, it detects only the fact that an attack occurred but does not indicate its type. Thus, the K-Means algorithm placed as a feature in the dataset to position in order to separate normal from anomalous traffic. In the second level, it is applied supervised algorithm to classify with the groups created by K-Means. The hybrid approach is designed to be able to classify new anomalies and maximize learning performance of the prediction ML model. The end to end implementation of ML models schematic in this research is illustrated in Figure 4.1.

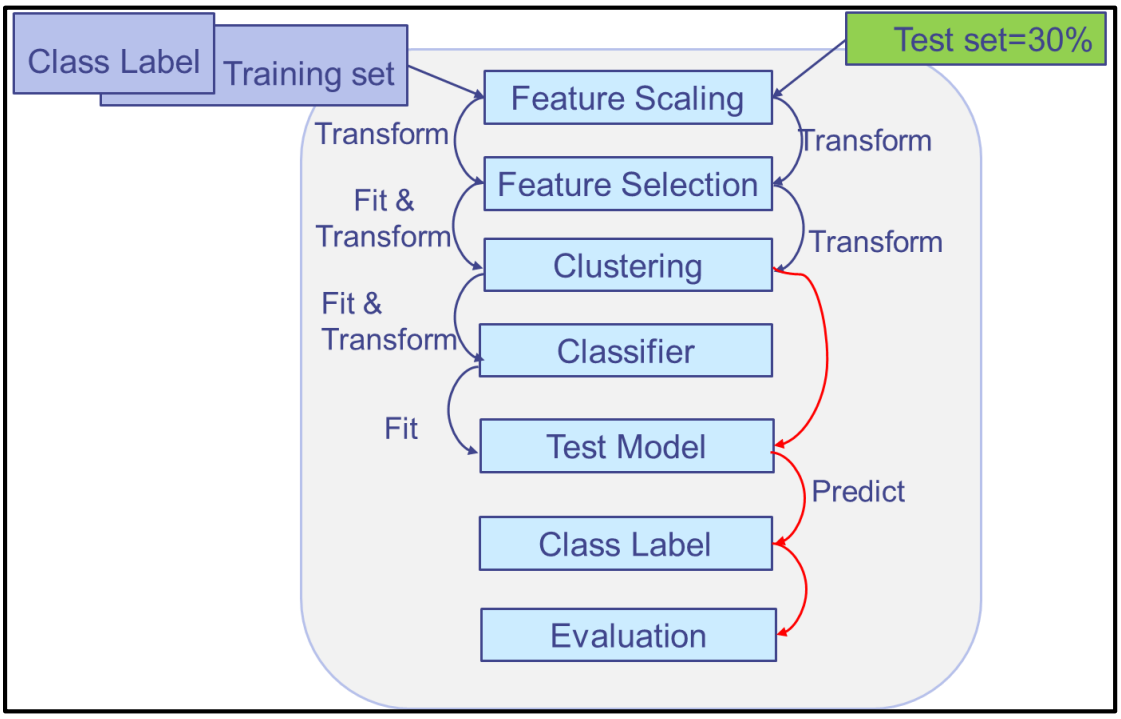


Figure 4.1 ML Model implementation scheme

4.2.1 Scenario I

4.2.1.1 PCA-DT Model Experimentation

Based on the KDD model framework the data mining process, dataset selection, preprocessing and feature selection techniques is applied then ML model training is made. In this scenario the researcher applied the PCA feature selection technique and clustering algorithm of K-Means algorithm and followed the supervised algorithm at the second level. In the k-Means clustering algorithm the cluster value for clustering is 2. K-Means receives as parameter two centroids as the aim of this algorithm was to classify the IDS dataset instance as Normal or Attack traffic in the second level, the DT classification algorithm was applied. The experimentation of this model is done by employing the 10-fold cross validation and the percentage split classification models. Table 4.3 shows summarizes of parameters for decision tree algorithm.

Table 4.3: Decision Tree Parameters [41]

Parameters	Description	Parameters Value
criterion	The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain	gini
splitter	The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split	best
max_depth	The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples	100
min_samples _split	The minimum number of samples required to split an internal node	2
random_state	Random state is the seed used by the random number generator	42

Based on this setup the classification model is built and the result found from this model is summarized in Table 4.4.

Table 4.4: Classification accuracy using PCA-DT model with 10-fold cross validation

Model	# of test dataset instances	Correctly classified	Incorrectly Classified	% correctly classified	% incorrect classification
PCA-DT	82263	82175	88	99.89302603	0.106973974

As shown in the resulting confusion matrix, the PCA-DT has classified 82175 dataset records correctly and 88 dataset records incorrectly. Thus, the PCA-DT model scored an accuracy of 99.89% while 0.11% of the records are incorrectly classified. The performance result of the model derived from confusion matrix is presented in Appendix D.

4.2.1.2 PCA-NN Model Experimentation

In this case all techniques are applied in the same fashion with PCA-DT model except the classifier algorithm NN is used instead of DT algorithm. The experimentation of this model is also done by employing the 10-fold cross validation and the percentage split classification models. Table 4.5 summarizes NN parameters with their values.

Table 4.5: NN Parameters [42]

Parameters	Description	Parameters Values
layers	An iteration sequence of each layer each as a sknn.mlp.Layer instance that contains its type, optional name, and any parameters required.	(30,30,30)
random_state	Seed for the initialization of the neural network parameters	0
n_stable	Number of iterations after which training should return when the validation error remains (near) constant.	10
valid_size	Ratio of the training data to be used for validation. 0.0 means no validation, and 1.0 would mean there's no training data! Common values are 0.1 or 0.25	0.1

Based on this setup the classification model is built and the result found from this model is summarized in Table 4.6.

Table 4.6: Classification accuracy using PCA-NN model with 10-fold cross validation

Model	**Number of test dataset instances**	**Correctly classified**	**Incorrectly Classified**	**% correctly classified**	**% incorrect classification**
PCA-NN	82263	80982	1281	98.44279931	1.557200686

As shown in the resulting confusion matrix, the PCA-NN has classified 80982 dataset records correctly while 1281 dataset records incorrectly. Thus, the PCA-NN model scored an accuracy of 98.44% while 1.56% of the records are incorrectly classified. The model performance result derived and confusion matrix of t is presented in Appendix D.

4.2.2 Scenario II

4.2.2.1 FI-DT Model Experimentation

In this scenario, feature importance is applied after the dataset is processed based on the KDD model. Feature importance is a feature selection technique which important features are extracted using random first classifier. After important features are selected, the training and test dataset is transformed. The training the ML model is made using the transformed training and test dataset then followed the clustering algorithm of K-Means algorithm. In the k-Means clustering algorithm the default value for clustering is 2. K-Means receives as parameter two centroids as the aim of this algorithm was to classify the IDS dataset instance as Normal or Attack traffic in the second level. In the second level, the DT classification algorithm was applied. The experimentation of this model is done by employing the 10-fold cross validation and the percentage split classification models. Table 4.7 presents classification accuracy of FI-DT model.

Table 4.7: Classification accuracy using FI-DT model with 10-fold cross validation

Model	# of test dataset instances	Correctly classified	Incorrectly Classified	% correctly classified	% incorrect classification
FI-DT	82263	81639	624	99.24145728	0.758542723

As shown in the resulting confusion matrix, the FI-DT has classified 81639 dataset records correctly and 624 dataset records incorrectly. Thus, the FI-DT model scored an accuracy of 99.24% while 0.76% of the records are incorrectly classified. The performance result of this model derived and the confusion matrix is presented in Appendix E.

4.2.2.2 FI-NN Model Experimentation

In this case all techniques are applied in the same fashion with FI-DT model except the classifier algorithm NN is used instead of DT algorithm. The experimentation of this model is also done by employing the 10-fold cross validation and the percentage split classification models. Table 4.8 summarizes the parameters with their values for neural network algorithm.

Table 4.8: Classification accuracy using FI-NN model

Model	# of test dataset instances	Correctly classified	Incorrectly Classified	% correctly classified	% incorrect classification
FI-NN	82263	81874	389	99.52712641	0.472873588

As shown in the resulting confusion matrix, the FI-NN has classified 80982 dataset records correctly while 1281 dataset records incorrectly. Thus, the FI-NN model scored an accuracy of 99.53% while 0.47% of the records are incorrectly classified. The performance result of this model derived and the confusion matrix is presented in Appendix E.

4.2.3 Scenario III

In this scenario, feature selection techniques are not employed but all pre-processing activities are made the same to the previous scenarios. The objective this scenario is to understand the impact of applying feature selection techniques during ML model construction. Based on this setup the classification models without feature selection technique is built and the result found from these models are summarized in Table 4.9 and 4.10.

Table 4.9: Classification accuracy using DT model

Model	# of test instances	Correctly classified	Incorrectly Classified	% of correctly classified	% of incorrect classification
DT	82263	81269	994	98.79168034	1.20832

As shown in the resulting confusion matrix, the DT has classified 81269 dataset records correctly and 994 dataset records incorrectly. Thus, the DT model scored an accuracy of 98.79% while 1.2% of the records are incorrectly classified. The performance result of this model derived and the confusion matrix is presented in Appendix F.

Table 4.10: Classification accuracy using NN model

Model	# of test instances	Correctly classified	Incorrectly Classified	% of correctly classified	% of incorrect classification
NN	82263	80065	2198	97.32808188	2.671918

`

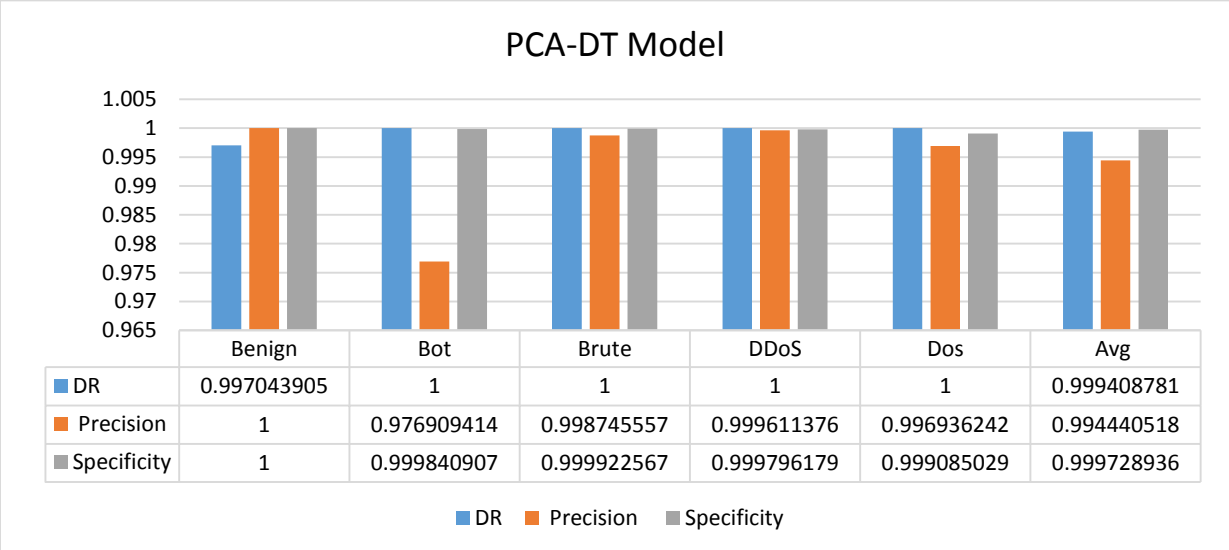
As shown in the resulting confusion matrix, the DT has classified 80065 dataset records correctly and 2198 dataset records incorrectly. Thus, the DT model scored an accuracy of 97.3% while 2.67% of the records are incorrectly classified. The performance result of this model derived and the confusion matrix is presented in Appendix F.

4.3 Analysis and Discussion of Results

In this section we show the results obtained with the different models. Comparing different classification techniques and selecting the best model for predicting the network intrusion detection is one of the objective of this study. Detail analysis of each models is made in the below sections.

4.3.1 Analysis over Scenario I

In this scenario, the models were used PCA as feature selection technique. The models are tested using the 10 set cross validation technique. With this test we evaluated the performance of the models against actual dataset entries. Specifically, in Figure 4.2 and 4.3, we show a comparative graph of the performance of the models from detection rate (DR), precision and specificity point of view, using five target classes corresponding to: Benign, Botnet, Brute force, DDoS and DoS. The result shows that the model PCA_DT has scored better performance than PCA_ NN based on the metrics. The performance result of this model derived and the confusion matrix is presented in Appendix D.



	Benign	Bot	Brute	DDoS	Dos	Avg
DR	0.997043905	1	1	1	1	0.999408781
Precision	1	0.976909414	0.998745557	0.999611376	0.996936242	0.994440518
Specificity	1	0.999840907	0.999922567	0.999796179	0.999085029	0.999728936

Figure 4.2 PCA-DT Model results derived from confusion matrix

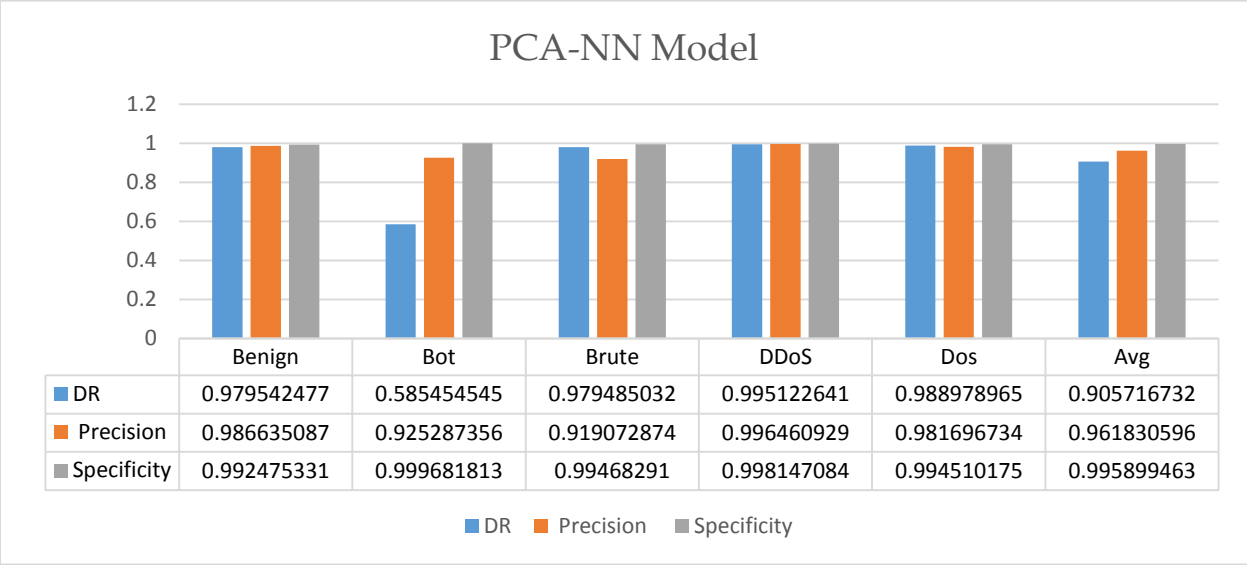


Figure 4.3: PCA-NN Model results derived from confusion matrix

4.3.2 Analysis over Scenario II

In this scenario, the models were used FI as feature selection technique. The models are tested using the 10 set cross validation technique. With this test we evaluated the performance of the models against actual dataset entries. Specifically, in Figure 4.4 and 4.5 we show a comparative graph of the performance of the models from detection rate(DR), precision and specificity point of view, using five target classes corresponding to: Benign, Botnet, Brute force, DDoS and DoS. The result shows that the model FI_DT and FI_NN has scored similar performance on the two criteria except the precision. FI_NN has scored better precision than FI_DT. The performance result of this model derived and the confusion matrix is presented in Appendix E.

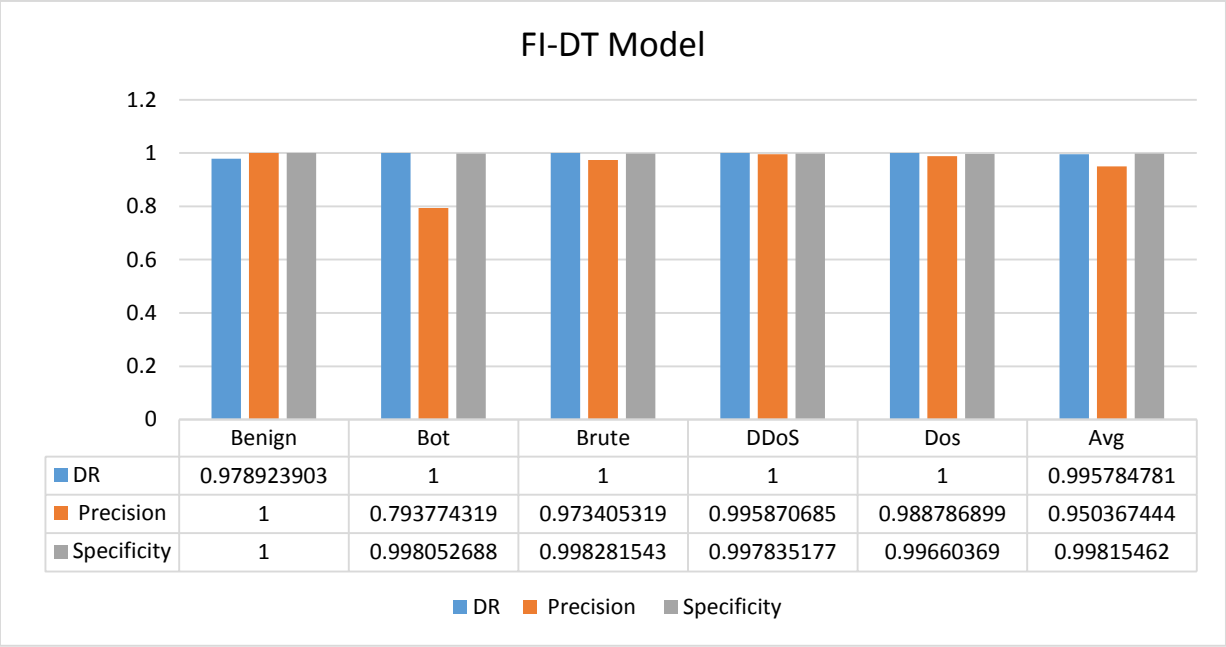


Figure 4.4: FI_DT Model results derived from confusion matrix

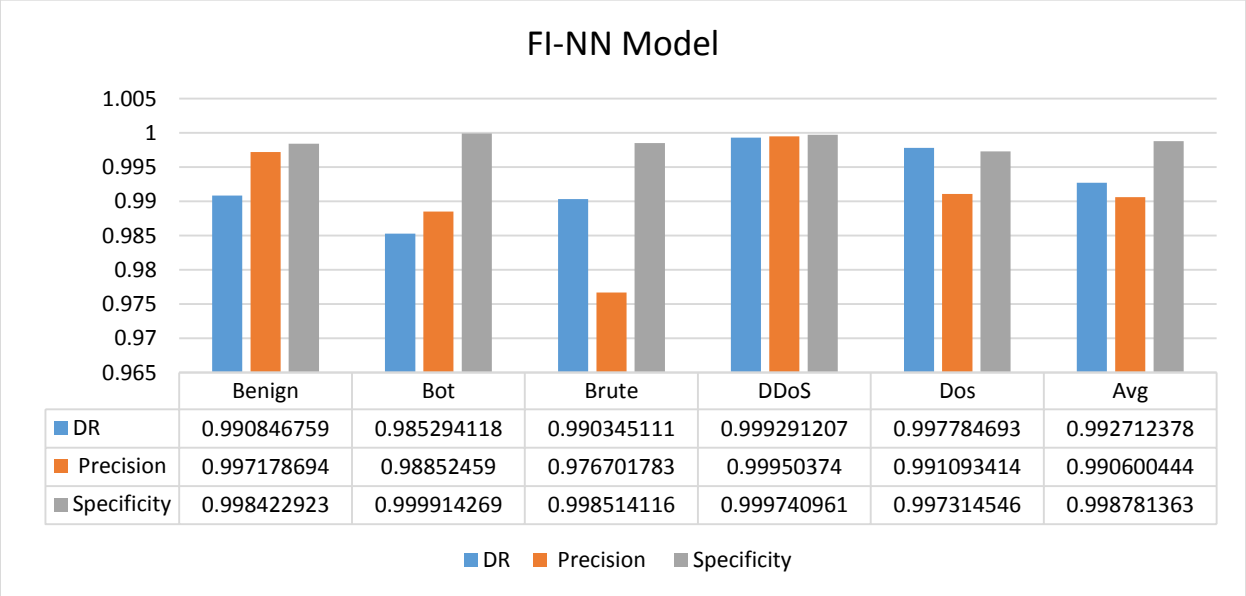
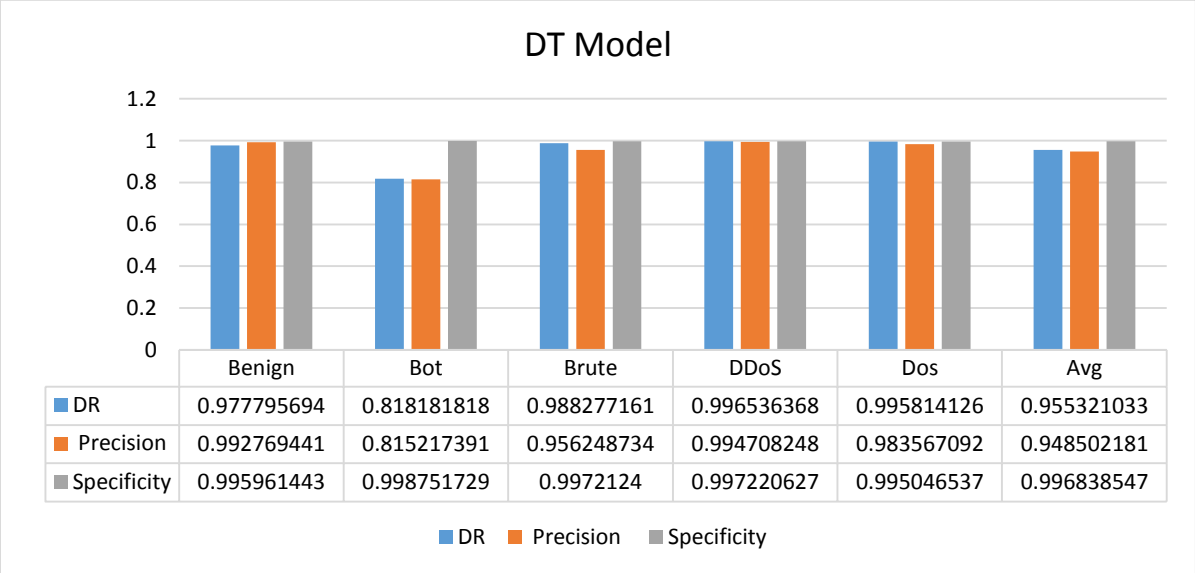


Figure 4.5: FI_NN Model results derived from confusion matrix

4.3.3 Analysis over Scenario III

In this scenario, the models were built without feature selection is applied to the dataset to evaluate the impact of feature selection in the performance of ML. The models are tested using the 10 set cross validation technique. With this test we evaluated the performance of the models against actual dataset entries. Specifically, in Figure 4.6 and 4.7, we show a comparative graph of the performance of the models from detection rate (DR), precision and specificity point of view, using five target classes corresponding to: Benign, Botnet, Brute force, DDoS and DoS. The result shows that the model DT has scored better performance than NN based on the three criteria. The performance result of this model derived and the confusion matrix is presented in Appendix D.



	Benign	Bot	Brute	DDoS	Dos	Avg
DR	0.977795694	0.818181818	0.988277161	0.996536368	0.995814126	0.955321033
Precision	0.992769441	0.815217391	0.956248734	0.994708248	0.983567092	0.948502181
Specificity	0.995961443	0.998751729	0.9972124	0.997220627	0.995046537	0.996838547

Figure 4.6: DT Model performance derived from confusion matrix

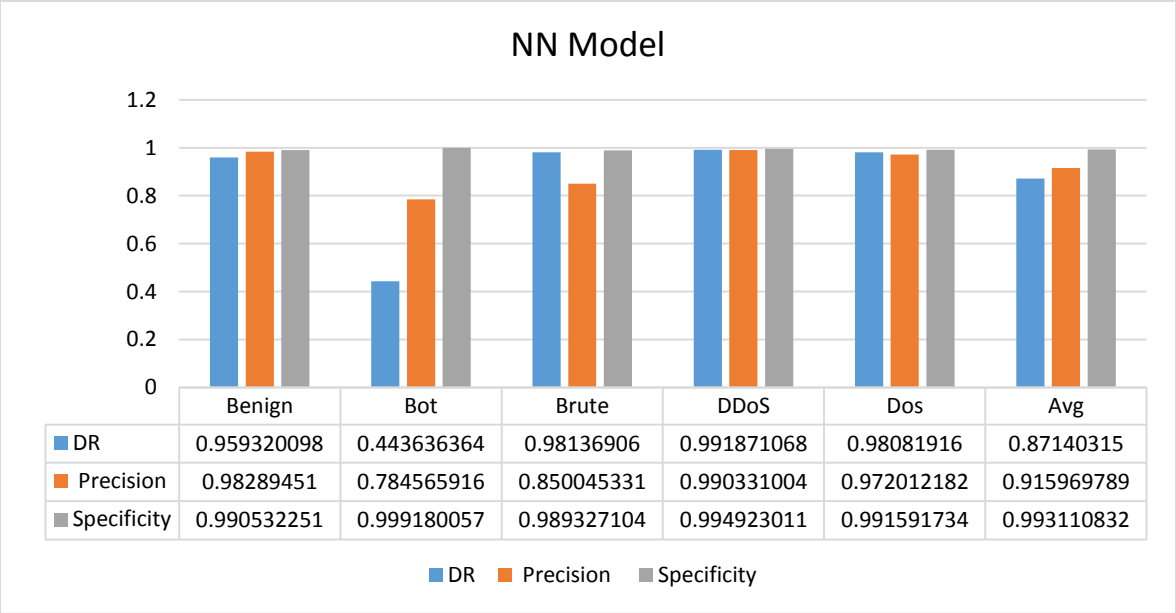


Figure 4.7: NN Model performance derived from confusion matrix

4.3.4 Comparison of Models

Comparing different classification ML models and selecting the best one for predicting the network intrusions is one of the goal of this study. In this section we present a final comparison of the models discussed by scenarios in the previous section. In this research, the decision trees algorithm and the Neural network classification algorithms were used for conducting experiments by applying two feature selection techniques for each. The models were trained over the training dataset after all preprocessing are made on the training dataset based on the KDD model. The performance of all models was evaluated with the same testing dataset using 10 gold cross validation. Summary of experimented result for the models built in this study are presented in Figure 4.8 below.

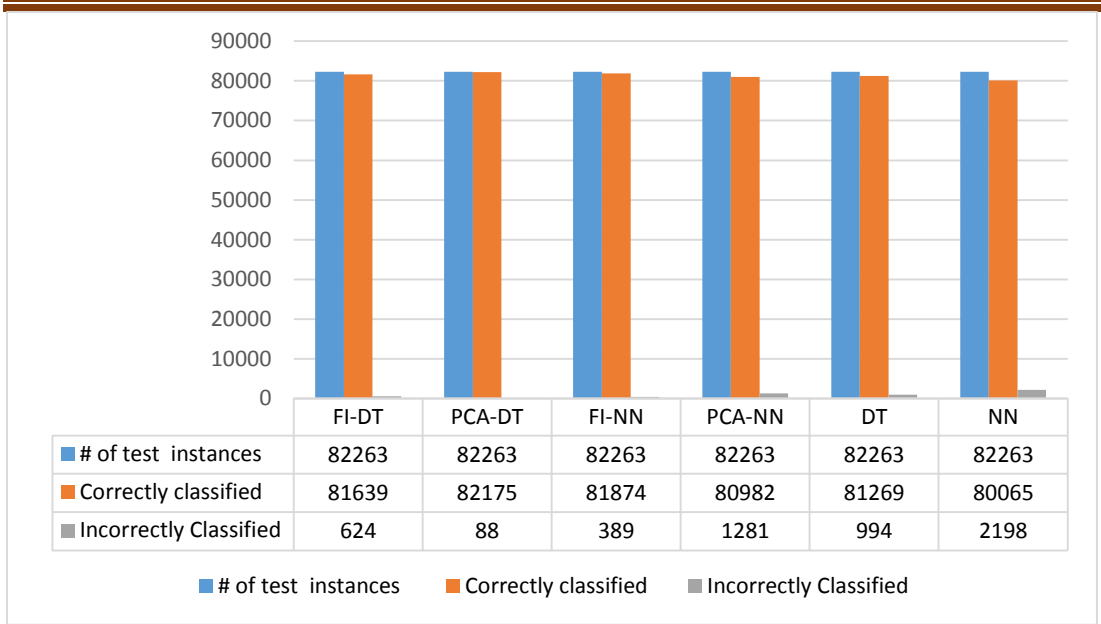


Figure 4.8: Global performance result of ML models

As we can see from the above Figure 4.8, the testing dataset instance allocated for all model is the same, 82263 which is 10% of the selected dataset for this study. All ML models has scored good in overall performance. Out of the 82263 testing records, PCA-DT has better classification performance than the remaining models which has predicted 82175 (99.89%) instances are correctly classified and 88 (0.1%) records are incorrectly classified. Whereas, PCA-NN has scored the lowest classification performance from all models which is 80065 (97.32%) instances are correctly classified and 2198 (2.67%) instances are incorrectly classified.

The models performance was evaluated from the error rate perspective as presented in Figure 4.9, Out of the 82263 testing records, PCA-DT has scored the lowest error rate which is 0.1% while NN has scored the highest error rate which is 2.67%.

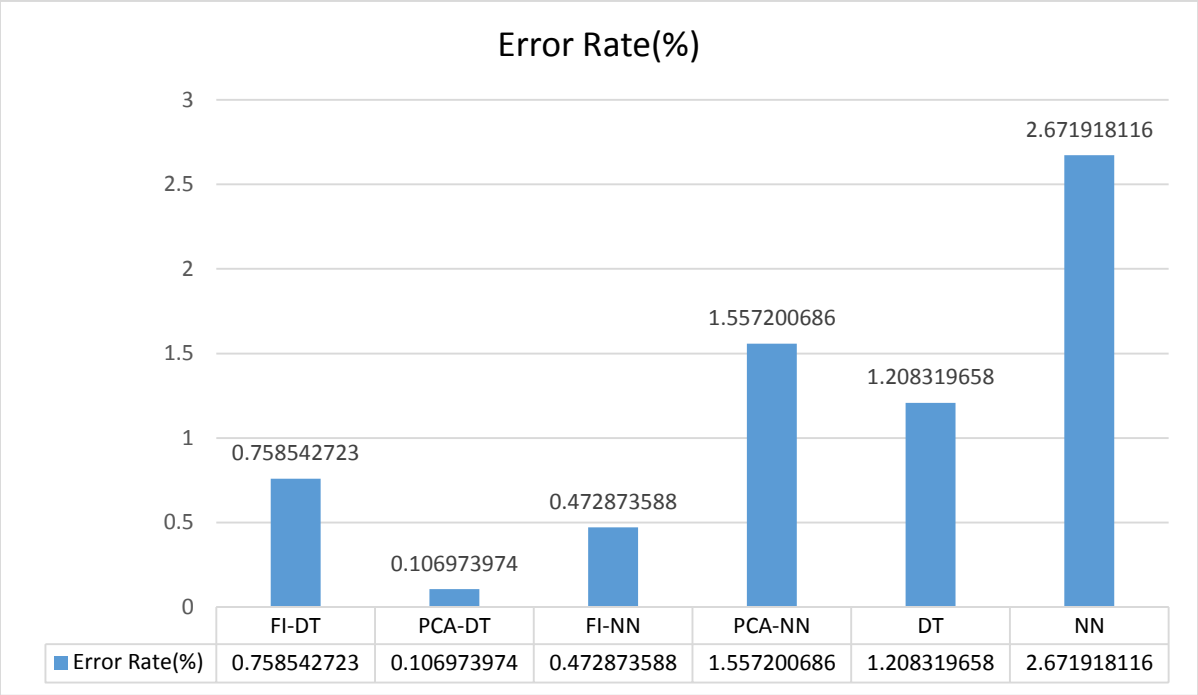


Figure 4.9: Comparison of ML models error rate

False positive rate, also known as False Alarm Rate (FAR) is another ML model performance metrics which measures the number of misclassified positive instances in relative to the total number of misclassified instances or the proportion of normal data is falsely detected as malicious. As shown in Figure 4.10, PCA-DT has lowest false positive rate (0.027%) which is the sign of good ML model. While the NN has the highest false positive rate (0.69%). In terms of FP rate, our PCA-DT is better than the other ML models.

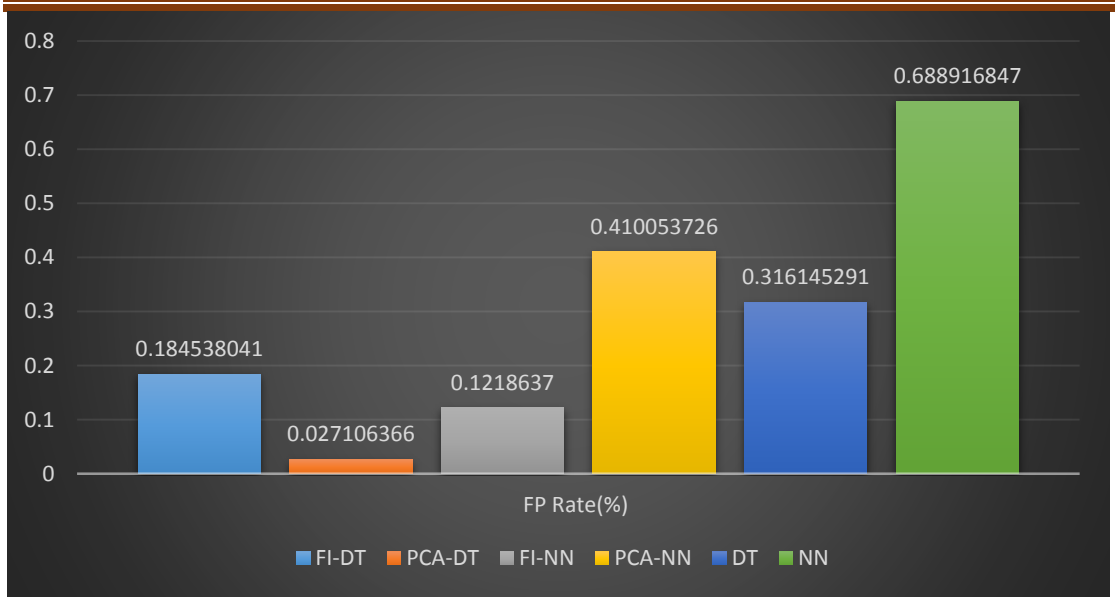
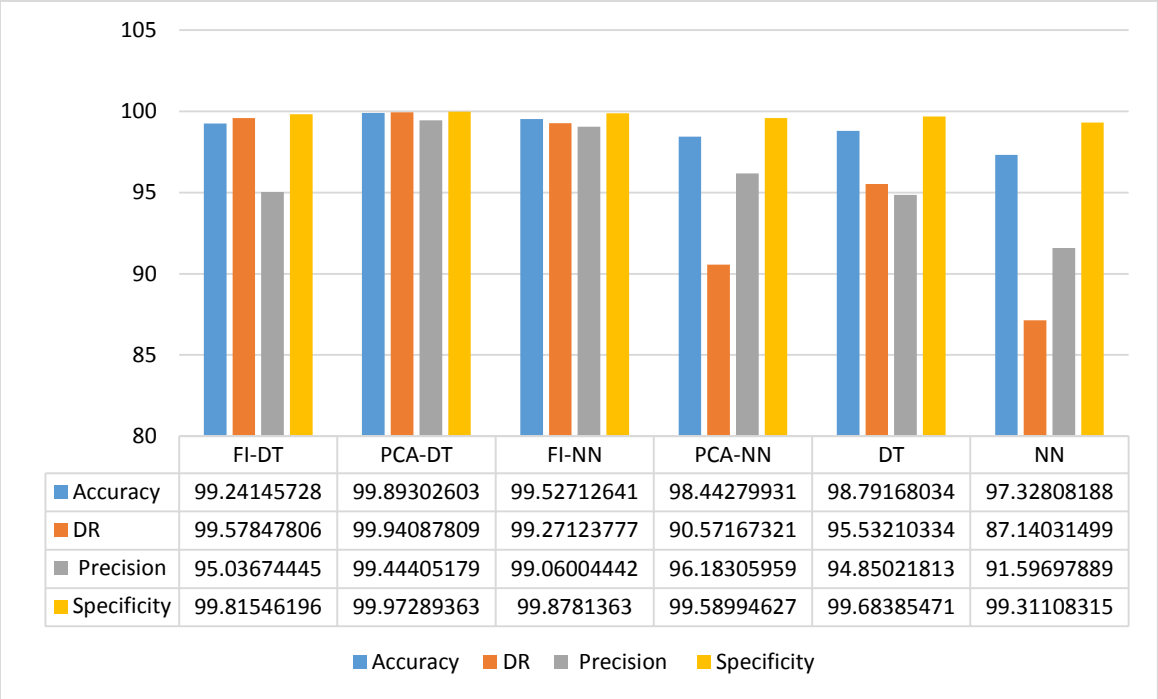


Figure 4.10: False Positive Rate (FPR) comparison of the DT and NN models

Figure 4.11 shows the global performance of the ML models. In terms of accuracy PCA-DT model has scored the best which was classified 82175 (99.89%) correctly out the total test dataset which is 82263. PCA-DT hybrid model is better than other classification models in terms of detection rate too; it has detected 99.94% out of the 52494 real attacks or positive samples. PCA- DT model has also scored 99.44% of precision. This shows the ML model is effective in identifying attacks and normal instances in a test dataset. In terms of Specificity, PCA-DT model scored slightly better than other models which has scored 99.97% which is the ML model is effective in identifying negative samples too. In technical term, the model will deny 0.03% negative sample instances. In general, the models FI-DT, PCA-DT, FI-NN and PCA-NN are the most balanced models in this scenario, since these models detected the greatest amount of attacks, classify the normal traffic in a better way, show better certainty when detecting attacks and a greater accuracy.



	FI-DT	PCA-DT	FI-NN	PCA-NN	DT	NN
Accuracy	99.24145728	99.89302603	99.52712641	98.44279931	98.79168034	97.32808188
DR	99.57847806	99.94087809	99.27123777	90.57167321	95.53210334	87.14031499
Precision	95.03674445	99.44405179	99.06004442	96.18305959	94.85021813	91.59697889
Specificity	99.81546196	99.97289363	99.8781363	99.58994627	99.68385471	99.31108315

Figure 4.11: Performance of the hybrid models over the testing dataset

To sum up, the results indicate that model PCA-DT is a good prototype hybrid ML model for the task of intrusion detection in computer networks because it shows good performance in this study. Additionally, it is the model that shows greater performance from all model performance evaluation standard metrics. The detailed result (accuracy and confusion matrix) of each ML models constructed in this thesis were presented in the Appendix D, E, and F. For comparison of these models, we have presented summarized experimentation results in the Table 19.

Table 4.11: Comparison of the confusion matrix result for ML Models

ML Models	Accuracy	Benign		Botnet		Brute		DDoS		DoS	
		DR	FP rate	DR	FP rate	DR	FP rate	DR	FP rate	DR	FP rate
FI-DT	99.24	0.979	0	1	0.002	1	0.0017	1	0.0021	1	0.0034
PCA-DT	99.89	0.997	0	1	0.00016	1	0	1	0.0002	1	0.0009
FI-NN	99.53	0.99	0.0016	0.985	0	0.99	0.001	0.999	0.0003	0.998	0.003
PCA-NN	98.44	0.98	0.0075	0.585	0.0003	0.979	0.005	0.995	0.0018	0.989	0.005
DT	98.79	0.978	0.0040	0.82	0.0012	0.99	0.0028	0.99	0.0028	0.99	0.0049
NN	97.3	0.96	0.0095	0.443	0.00082	0.98	0.011	0.99	0.0051	0.98	0.0084

4.4 Sample IDS Decision Rules

In this section, sample rules are presented which are findings of this study that the ML models generates interesting rules that are crucial for intrusion detection in the computer network security. These rules of IDS are derived from the model PCA-DT which has achieved better performance.

Some of the rules generated from the selected model are the following:

Rule 1: Bwd Packet Length Min <= 11 and Destination Port <= 20 and Bwd Packet Length Min <= 2 = BENIGN

Rule 2: Bwd Packet Length Min <= 11 and Destination Port <= 20 Bwd Packet Length Min > 2 = DDoS

Rule 3: Destination Port > 20 and Init_Win_bytes_backward <= 14600 and min_seg_size_forward <= 24= BENIGN

Rule 4: Destination Port <= 443 and Bwd Packet Length Min <= 11 and Destination Port > 22 and Destination Port <= 427 and Max Packet Length <= 0 and Init_Win_bytes_forward <= 233= DoS

Rule 5: Init_Win_bytes_backward > 234 and Init_Win_bytes_backward <= 235 and Max Packet Length <= 2171 and Bwd Packet Length Mean <= 26.6 and Init_Win_bytes_forward <= 246= Brute

Rule 6: Bwd Packet Length Min > 2 and Init_Win_bytes_forward > 235 and Init_Win_bytes_forward <= 238 and Init_Win_bytes_backward >255=Bot

Rule 7: Init_Win_bytes_forward > 1023 and min_seg_size_forward <= 20 and Bwd Packet Length Mean <= 6.2 and Destination Port <= 726 and Init_Win_bytes_forward <= 1536=DDoS

Rule 8: Destination Port > 726 and Init_Win_bytes_forward <= 1496 and Init_Win_bytes_forward > 1106 and Init_Win_bytes_backward <= 16404=Bot

Rule 9: min_seg_size_forward > 20 and Bwd Packet Length Min <= 2 and Destination Port <= 445= DDoS

Rule 10: Destination Port > 443 and Init_Win_bytes_forward <= 1023 and Bwd Packet Length Min <= 2 and Bwd Packet Length Mean > 49.9 and Max Packet Length > 7240= DoS (10.0)

Rule 11: Destination Port > 443 and Init_Win_bytes_forward <= 1023 and Bwd Packet Length Min <= 2 and Bwd Packet Length Mean <= 49.9= BENIGN

Rule 12: Destination Port > 22 and Destination Port <= 427 and Init_Win_bytes_backward <= 234 and Max Packet Length <= 0 and Init_Win_bytes_backward <= 4 and Init_Win_bytes_forward <= 233=DoS

Rule 13: Init_Win_bytes_backward > 236 and Max Packet Length <= 2 and Init_Win_bytes_backward <= 28560 and Init_Win_bytes_forward > 268 and Init_Win_bytes_forward <= 288= Brute

Rule 14: Bwd Packet Length Min > 11 and Bwd Packet Length Min > 517 and Destination Port <= 84 and min_seg_size_forward <= 24 Init_Win_bytes_backward <= 232=DDoS

Rule 15: Bwd Packet Length Min > 11 and Bwd Packet Length Min > 517 and Destination Port <= 84 and min_seg_size_forward <= 24 and Init_Win_bytes_backward > 232=DoS

Chapter 5

Conclusion and Future Works

The revolution of IT has brought an important transformation in enterprises from different aspects, increasing the reliance on technology to gain competitive advantage and it has promoted the emergence of cloud-based systems, internet of things (IoT) and big data. However, security issues have been one of the most critical problem for enterprises as attackers also evolved dynamically. Because of the dynamic change of the IT and sophistication attacks in computer networks, there should be a tool to protect from such security issues. ML techniques are one of the technologies that are applied for intrusion detection systems.

In this study, different hybrid ML models are developed using ML techniques. It combines the supervised, unsupervised and feature selection algorithms. The aim of applying hybrid ML model is to detect attacks which have previously known or labeled and new emerging attacks or not labeled. To accomplish this study, the CIC-IDS2017 dataset is used in order to evaluate the performance of the proposed models which is taken from University of New Brunswick institute (Canada Institute of Cyber Security).

The models use the latest research trends in the subject as presented in the literature review part, using the DT and NN algorithms for supervised learning, the K-Means unsupervised learning algorithm and the PCA and FI feature selection techniques. The KDD process model has been followed from the data selection up to the discussion of results. The models are implemented in python programming language by using the CIC-IDS2017 dataset.

The results shown in the previous chapter indicate that DT shows better performance than the NN and PCA feature selection technique is better than FI feature selection technique; Model PCA-DT is the better ML model since it shown the most balanced results in all performance parameters. Finally, Feature selection is an integral component of knowledge discovery and machine learning which helps build robust and cost-effective learning models for the extraction of interesting hidden patterns by selecting a subset of relevant features. Hybrid based ML models are a better approach because they allow the detection of known and unknown attacks and the result shown that feature selection techniques play significant role in the performance of ML models. The results confirm the effectiveness of our proposed methods and it can contribute towards in improving the computer network security.

Finally, we recommend ethio telecom to deploy hybrid ML based IDS to improve its security defense as the current security is signature based IDS.

References

[1] D. K. Bhattacharyya and J. K. Kalita, *Network anomaly detection: A machine learning perspective*: Crc Press, 2013.

[2] D. Perez, M. A. Astor, D. P. Abreu, and E. Scalise, "Intrusion detection in computer networks using hybrid machine learning techniques," in *Computer Conference (CLEI), 2017 XLIII Latin American*, 2017, pp. 1-10.

[3] J. N. Keerthi Latha M R, "Learn About Intrusion Detection and Prevention," *Juniper* 2016.

[4] I. T. R. X. 805, "Security architecture for systems providing end-to-end communications," ed, 2003.

[5] ITU, "Global Cybersecurity Index (GCI)," 2017.

[6] S. Morgan. (2017, 05/14). *Cybersecurity Business Report*. Available: https://www.csoonline.com/article/3153707/security/top-5-cybersecurity-facts-figures-and-statistics.html

[7] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, 2009, pp. 1-6.

[8] R. Tewatia and A. Mishra, "Introduction to intrusion detection system review," *International journal of scientific & technology research,* vol. 4, pp. 219-223, 2015.

[9] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications,* vol. 36, pp. 16-24, 2013.

[10] CISCO, "cisco 2018 anual cybersecurity report " 2018.

[11] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The KDD process for extracting useful knowledge from volumes of data," *Communications of the ACM*, vol. 39, pp. 27-34, 1996.

[12] P. Julio and K. Adem, *Data mining and knowledge discovery in real life applications*, 2011.

[13] R. Wirth and J. Hipp, "CRISP-DM: Towards a standard process model for data mining," in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, 2000, pp. 29-39.

[14] M. Jabbar, R. Aluvalu, and S. Reddy, "Cluster Based Ensemble Classification for Intrusion Detection System," in *Proceedings of the 9th International Conference on Machine Learning and Computing*, 2017, pp. 253-257.

[15] K. S. Elekar, "Combination of data mining techniques for intrusion detection system," in *Computer, Communication and Control (IC4), 2015 International Conference on*, 2015, pp. 1-5.

[16] H. Mohamad Tahir, W. Hasan, A. Md Said, N. H. Zakaria, N. Katuk, N. F. Kabir, *et al.*, "Hybrid machine learning technique for intrusion detection system," 2015.

[17] B. Aslahi-Shahri, R. Rahmani, M. Chizari, A. Maralani, M. Eslami, M. Golkar, *et al.*, "A hybrid method consisting of GA and SVM for intrusion detection system," *Neural computing and applications*, vol. 27, pp. 1669-1676, 2016.

[18] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Expert Systems with Applications*, vol. 41, pp. 1690-1700, 2014.

[19] K. Kumar, "An efficient network intrusion detection system based on fuzzy C-means and support vector machine," in *Computer, Electrical & Communication Engineering (ICCECE), 2016 International Conference on*, 2016, pp. 1-6.

[20] L. Dhanabal and S. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, pp. 446-452, 2015.

[21] S. K. Sahu, S. Sarangi, and S. K. Jena, "A detail analysis on intrusion detection datasets," in *Advance Computing Conference (IACC), 2014 IEEE International*, 2014, pp. 1348-1353.

[22] A. Özgür and H. Erdem, "A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015," *PeerJ PrePrints*, 2016.

[23] S. Revathi and A. Malathi, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," *International Journal of Engineering Research and Technology. ESRSA Publications*, 2013.

[24] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "An Evaluation Framework for Intrusion Detection Dataset," in *Information Science and Security (ICISS), 2016 International Conference on*, 2016, pp. 1-6.

[25] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a Reliable Intrusion Detection Benchmark Dataset," *Software Networking*, vol. 2018, pp. 177-200, 2018.

[26] M. Usama, J. Qadir, A. Raza, H. Arif, K.-L. A. Yau, Y. Elkhatib, *et al.*, "Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges," *arXiv preprint arXiv:1709.06599*, 2017.

[27] M. Jianliang, S. Haikun, and B. Ling, "The application on intrusion detection based on k-means cluster algorithm," in *Information Technology and Applications, 2009. IFITA'09. International Forum on*, 2009, pp. 150-152.

[28] R. Chitrakar and H. Chuanhe, "Anomaly detection using Support Vector Machine classification with k-Medoids clustering," in *Internet (AH-ICI), 2012 Third Asian Himalayas International Conference on*, 2012, pp. 1-5.

[29] C. C. Aggarwal, *Data mining: the textbook*: Springer, 2015.

[30] T. Dagne, "Constructing a Semi-Supervised Model for Network Intrusion Detection," Addis Ababa university, 2012.

[31] (2017, 05/20/2018). *Intrusion Detection Evaluation Dataset (CICIDS2017)*. Available: http://www.unb.ca/cic/datasets/IDS2017.html

[32] J. S. Rojas, Á. R. Gallón, and J. C. Corrales, "Personalized Service Degradation Policies on OTT Applications Based on the Consumption Behavior of Users," in *International Conference on Computational Science and Its Applications*, 2018, pp. 543-557.

[33] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," *Software Networking*, vol. 1, pp. 177-200, 2017.

[34] E. N. Stankova, E. Ismailova, and I. A. Grechko, "Algorithm for Processing the Results of Cloud Convection Simulation Using the Methods of Machine Learning," in *International Conference on Computational Science and Its Applications*, 2018, pp. 149-159.

[35] S. Raschka, *Python machine learning*: Packt Publishing Ltd, 2015.

[36] (2015, 22/2018). *Seven Techniques for Data Dimensionality Reduction*. Available: https://www.kdnuggets.com/2015/05/7-methods-data-dimensionality-reduction.html

[37] F. Abdel-Fattah, F. Dahalin, and S. Jusoh, "Distributed and cooperative hierarchical intrusion detection on MANETs," *International Journal of Computer Applications*, vol. 12, 2010.

[38] (2018, Aug 3). *Generalized Confusion Matrix for Multiple Classes*. Available: https://www.researchgate.net/publication/310799885/download

[39] Y. Jiao and P. Du, "Performance measures in evaluating machine learning based bioinformatics predictors for classifications," *Quantitative Biology*, vol. 4, pp. 320-330, 2016.

[40] Gregory Piatetsky. (2018, 20/08). *Top Software for Analytics, Data Science, Machine Learning in 2018: Trends and Analysis*. Available: https://www.kdnuggets.com/2018/05/poll-tools-analytics-data-science-machine-learning-results.html

[41] (2018, Aug 10). *sklearn.tree.DecisionTreeClassifier*. Available: http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[42] (2018, Aug 6). *sklearn.neural_network.MLPClassifier*. Available: http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

APPENDIXES

Appendix A: Description of Dataset Features

Description of CIC-IDS2017 Intrusion Detection Dataset Features[32]

Feature Name	Description
Flow Duration	Flow duration
Total Fwd Packets	Total packets in the forward direction
Total Backward Packets	Total packets in the backward direction
Total Length of Fwd Packets	Total size of packet in forward direction
Total Length of Bwd Packets	Total size of packet in backward direction
Fwd Packet Length Max	Maximum size of packet in forward direction
Fwd Packet Length Min	Minimum size of packet in forward direction
Fwd Packet Length Mean	Average size of packet in forward direction
Fwd Packet Length Std	Standard deviation size of packet in forward direction
Bwd Packet Length Max	Maximum size of packet in backward direction
Bwd Packet Length Min	Minimum size of packet in backward direction
Bwd Packet Length Mean	Mean size of packet in backward direction
Bwd Packet Length Std	Standard deviation size of packet in backward direction
Flow Bytes/s	flow byte rate that is number of packets transferred per second
Flow Packets/s	flow packets rate that is number of packets transferred per second
Flow IAT Mean	Average time between two flows
Flow IAT Std	Standard deviation time two flows
Flow IAT Max	Maximum time between two flows
Flow IAT Min	Minimum time between two flows
Fwd IAT Total	Total time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction

Feature Name	Description
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Bwd IAT Total	Total time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Fwd PSH Flags	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd Header Length	Total bytes used for headers in the forward direction
Bwd Header Length	Total bytes used for headers in the forward direction
Fwd Packets/s	Number of forward packets per second
Bwd Packets/s	Number of backward packets per second
Min Packet Length	Minimum length of a flow
Max Packet Length	Maximum length of a flow
Packet Length Mean	Mean length of a flow
Packet Length Std	Standard deviation length of a flow
Packet Length Variance	Minimum inter-arrival time of packet
FIN Flag Count	Number of packets with FIN
SYN Flag Count	Number of packets with SYN
RST Flag Count	Number of packets with RST
PSH Flag Count	Number of packets with PUSH
ACK Flag Count	Number of packets with ACK
URG Flag Count	Number of packets with URG

Feature Name	Description
CWE Flag Count	Number of packets with CWE
ECE Flag Count	Number of packets with ECE
Down/Up Ratio	Download and upload ratio
Average Packet Size	Average size of packet
Avg Fwd Segment Size	Average size observed in the forward direction
Avg Bwd Segment Size	Average size observed in the backward direction
Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in the forward direction
Fwd Avg Packets/Bulk	Average number of packets bulk rate in the forward direction
Fwd Avg Bulk Rate	Average number of bulk rate in the forward direction
Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in the backward direction
Bwd Avg Packets/Bulk	Average number of packets bulk rate in the backward direction
Bwd Avg Bulk Rate	Average number of bulk rate in the backward direction
Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
Init_Win_bytes_forward	Number of bytes sent in initial window in the forward direction
Init_Win_bytes_backward	# of bytes sent in initial window in the backward direction
act_data_pkt_fwd	# of packets with at least 1 byte of TCP data payload in the forward direction
min_seg_size_forward	Minimum segment size observed in the forward direction
Active Mean	Mean time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Min	Minimum time a flow was active before becoming idle

Feature Name	Description
Idle Mean	Mean time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Min	Minimum time a flow was idle before becoming active

Appendix B: Main Statistical Characteristics of Dataset

	Source Port	Destination Port	Protocol	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packe Length Mi
count	590286.000000	590286.000000	590286.000000	5.902860e+05	590286.000000	590286.000000	590286.000000	5.902860e+05	590286.000000	590286.00000
mean	42683.544719	7048.820509	9.720269	1.567735e+07	10.007712	11.315225	453.849354	1.801146e+04	162.406552	15.82078
std	21715.592936	17547.898105	5.209405	3.441685e+07	753.622349	1018.789164	5080.980076	2.282183e+06	439.293015	34.87696
min	0.000000	0.000000	0.000000	-4.000000e+00	1.000000	0.000000	0.000000	0.000000e+00	0.000000	0.00000
25%	35938.000000	53.000000	6.000000	1.890000e+02	2.000000	1.000000	2.000000	0.000000e+00	2.000000	0.00000
50%	51789.000000	80.000000	6.000000	4.737000e+04	2.000000	2.000000	62.000000	1.410000e+02	37.000000	0.00000
75%	58498.000000	443.000000	17.000000	5.027245e+06	5.000000	5.000000	286.000000	1.958000e+03	181.000000	36.00000
max	65535.000000	65505.000000	17.000000	1.200000e+08	206446.000000	276072.000000	2428415.000000	6.270000e+08	24820.000000	2065.00000

8 rows × 78 columns

act_data_pkt_fwd	min_seg_size_forward	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min
274208.000000	274208.000000	2.742080e+05	2.742080e+05	2.742080e+05	2.742080e+05	2.742080e+05	2.742080e+05	2.742080e+05	2.742080e+05
2.306085	29.969107	1.581683e+05	5.906602e+04	2.215847e+05	1.187317e+05	1.068208e+07	8.156398e+05	1.143816e+07	1.006549e+07
29.739517	7.631589	9.991205e+05	5.651892e+05	1.313890e+06	9.061473e+05	2.755748e+07	5.673858e+06	2.857437e+07	2.728351e+07
0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000	20.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
0.000000	32.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
1.000000	40.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
6697.000000	56.000000	1.050000e+08	5.040000e+07	1.050000e+08	1.050000e+08	1.200000e+08	7.310000e+07	1.200000e+08	1.200000e+08

Appendix C: Sample Scaled Dataset

```
[[6.19821469e-01 1.22344737e-03 3.52941176e-01 4.18422005e-02
  3.82760468e-04 2.59713277e-04 1.12494399e-03 1.43432716e-02
  0.00000000e+00 6.04281768e-02 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 4.78400230e-04
  2.72695399e-04 0.00000000e+00 4.23045931e-02 0.00000000e+00
  0.00000000e+00 0.00000000e+00 1.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 3.82760468e-04 1.12494399e-03
  2.59713277e-04 2.90679806e-05 4.45571899e-01 3.72314453e-03
  2.98641183e-04 5.71428571e-01 2.90946502e-05 1.97221821e-02
  2.57517453e-02 4.87544177e-02 5.57587013e-02 4.18422413e-03
  1.85168732e-02 4.13804913e-02 2.58333308e-07 4.61675000e-04
  9.23350000e-05 2.10930144e-04 3.48858333e-04 2.24999985e-07
  4.18410667e-02 1.04602667e-02 2.99752949e-02 4.17094417e-02
  8.77500000e-06 3.98322346e-07 4.97902933e-07 7.78680000e-02
  6.74180838e-02 4.53999556e-03 8.26694783e-02 1.97221821e-02
  4.87544177e-02 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [6.19821469e-01 1.22344737e-03 3.52941176e-01 5.91667145e-07
  0.00000000e+00 5.19426553e-05 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 7.65440367e-05
  5.19419808e-05 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 1.00000000e+00
  1.00000000e+00 0.00000000e+00 0.00000000e+00 1.25000000e-01
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  5.19426553e-05 0.00000000e+00 3.87573242e-03 3.72314453e-03
  0.00000000e+00 5.71428571e-01 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 5.91666662e-07
  0.00000000e+00 5.91666662e-07 6.83333265e-07 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 6.66666622e-08
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 4.76190476e-03 7.14285714e-03 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [6.19851987e-01 1.22344737e-03 3.52941176e-01 4.18684672e-02
  3.82760468e-04 2.59713277e-04 1.12494399e-03 1.43432716e-02
  0.00000000e+00 6.04281768e-02 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 4.78400230e-04
  2.72695399e-04 0.00000000e+00 4.23045931e-02 0.00000000e+00
  0.00000000e+00 0.00000000e+00 1.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 3.82760468e-04 1.12494399e-03
  2.59713277e-04 2.90679806e-05 4.45571899e-01 3.72314453e-03
  2.98641183e-04 5.71428571e-01 2.90946502e-05 1.97221821e-02
  2.57517453e-02 4.87544177e-02 5.57587013e-02 4.18685080e-03
```

```
 1.85082290e-02 4.13657080e-02 2.58333308e-07 5.02725000e-04
 1.00545000e-04 2.14411952e-04 3.53675000e-04 2.24999985e-07
 4.18673000e-02 1.04668250e-02 2.99556156e-02 4.16954083e-02
 7.90000000e-06 3.98072454e-07 4.97590567e-07 7.78680000e-02
 6.74180838e-02 4.53999556e-03 8.26694783e-02 1.97221821e-02
 4.87544177e-02 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00]
[6.19851987e-01 1.22344737e-03 3.52941176e-01 4.41667024e-07
 0.00000000e+00 5.19426553e-05 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 7.65440367e-05
 5.19419808e-05 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 1.00000000e+00
 1.00000000e+00 0.00000000e+00 0.00000000e+00 1.25000000e-01
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 5.19426553e-05 0.00000000e+00 3.87573242e-03 3.72314453e-03
 0.00000000e+00 5.71428571e-01 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 4.41666663e-07
 0.00000000e+00 4.41666663e-07 5.33333280e-07 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 6.66666622e-08
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 6.41025641e-03 9.61538461e-03 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00]
[6.71900511e-01 6.22428849e-03 3.52941176e-01 7.66667286e-07
 0.00000000e+00 5.19426553e-05 3.20040965e-06 8.05801773e-05
 8.60215054e-04 3.45303867e-04 3.02571861e-03 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 5.74080276e-05
 3.24637380e-05 1.54559505e-03 2.41740532e-04 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 1.25000000e-01
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 3.20040965e-06
 5.19426553e-05 1.23343624e-07 1.56402588e-02 1.52587891e-05
 0.00000000e+00 4.28571429e-01 1.23456790e-07 3.36651758e-04
 0.00000000e+00 1.03439359e-03 0.00000000e+00 7.66666660e-07
 0.00000000e+00 7.66666660e-07 8.58333248e-07 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 6.66666622e-08
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 3.66300366e-03 5.49450550e-03 1.47128956e-03
 4.90423833e-04 2.40240240e-07 2.14776632e-03 3.36651758e-04
 1.03439359e-03 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00]]
```

Appendix D: PCA_DT and PCA_NN models Performance Result

Table D-1: Confusion Matrix using DT algorithm with PCA feature selection

	BENIGN	Bot	Brute	DDoS	DoS	Sum
BENIGN	29681	13	6	11	58	**29769**
Bot	0	550	0	0	0	**550**
Brute	0	0	4777	0	0	**4777**
DDoS	0	0	0	28294	0	**28294**
DoS	0	0	0	0	18873	**18873**
Sum	**29681**	**563**	**4783**	**28305**	**18931**	**82263**

Table D-2: Detailed Accuracy by Class using DT algorithm with PCA feature selection

	TP	FN	FP	TN	DR	FP Rate	Precision	Specificity	support
Benign	29681	88	0	52494	0.997044	0	1	1	29769
Bot	550	0	13	81700	1	0.0001591	0.976909	0.999841	550
Brute	4777	0	6	77480	1	7.743E-05	0.998746	0.999923	4777
DDoS	28294	0	11	53958	1	0.0002038	0.999611	0.999796	28294
Dos	18873	0	58	63332	1	0.000915	0.996936	0.999085	18873
avg / total					0.999409	0.0002711	0.994441	0.999729	82263

Table D-3: Confusion Matrix using NN algorithm with PCA feature selection

	BENIGN	Bot	Brute	DDoS	DoS	Sum
BENIGN	29160	26	323	33	227	**29769**
Bot	228	322	0	0	0	**550**
Brute	65	0	4679	0	33	**4777**
DDoS	49	0	1	28156	88	**28294**
DoS	53	0	88	67	18665	**18873**
sum	**29555**	**348**	**5091**	**28256**	**19013**	**82263**

Table D-4: Detailed Accuracy by Class using NN algorithm with PCA feature selection

	TP	FN	FP	TN	DR	FP Rate	precision	Specificity	support
Benign	29160	609	395	52099	0.979542	0.007525	0.986635	0.992475	29769
Bot	322	228	26	81687	0.585455	0.000318	0.925287	0.999682	550
Brute	4679	98	412	77074	0.979485	0.005317	0.919073	0.994683	4777
DDoS	28156	138	100	53869	0.995123	0.001853	0.996461	0.998147	28294
Dos	18665	208	348	63042	0.988979	0.00549	0.981697	0.99451	18873
					0.905717	0.004101	0.961831	0.995899	82263

Appendix E: FI_DT and FI_NN Models Performance Result

Table E-1: Confusion Matrix using DT algorithm with FI feature selection

	BENIGN	Bot	Brute	DDoS	DoS	Sum
BENIGN	28983	159	133	117	215	**29607**
Bot	0	612	0	0	0	**612**
Brute	0	0	4868	0	0	**4868**
DDoS	0			28217	0	**28217**
DoS	0	0	0	0	18959	**18959**
Sum	**28983**	**771**	**5001**	**28334**	**19174**	**82263**

Table E-2: Detailed Accuracy by Class using DT algorithm with FI feature selection

	TP	FN	FP	TN	DR	FP Rate	Precision	Specificity	Sum
Benign	28983	624	0	52656	0.97892	0	1	1	29607
Bot	612	0	159	81492	1	0.00195	0.79377	0.99805	612
Brute	4868	0	133	77262	1	0.00172	0.97341	0.99828	4868
DDoS	28217	0	117	53929	1	0.00216	0.99587	0.99784	28217
Dos	18959	0	215	63089	1	0.0034	0.98879	0.9966	18959
Avg / total					0.99578	0.00185	0.95037	0.99815	82263

Table E-3: Confusion Matrix using NN algorithm with FI feature selection

	BENIGN	Bot	Brute	DDoS	DoS	Sum
BENIGN	29336	7	109	12	143	**29607**
Bot	9	603	0	0	0	**612**
Brute	20	0	4821	0	27	**4868**
DDoS	19	0	1	28197	0	**28217**
DoS	35	0	5	2	18917	**18959**
Sum	**29419**	**610**	**4936**	**28211**	**19087**	**82263**

Table E-4: Detailed Accuracy by Class using NN algorithm with FI feature selection

	TP	FN	FP	TN	DR	FP Rate	precision	Specificity	Support
Benign	29336	271	83	52546	0.990847	0.001577	0.997179	0.9984229	29607
Bot	603	9	7	81644	0.985294	8.57E-05	0.988525	0.9999143	612
Brute	4821	47	115	77280	0.990345	0.001486	0.976702	0.9985141	4868
DDoS	28197	20	14	54032	0.999291	0.000259	0.999504	0.999741	28217
Dos	18917	42	170	63134	0.997785	0.002685	0.991093	0.9973145	18959
Avg /Total					0.992712	0.001219	0.9906	0.9987814	82263

Appendix F: DT and NN Models Performance Result

Table F-1: Confusion Matrix using DT algorithm without feature selection

	BENIGN	Bot	Brute	DDoS	DoS	Sum
BENIGN	29108	84	193	117	267	29769
Bot	30	450	23	0	47	550
Brute	16	7	4721	33	0	4777
DDoS	87	11	0	28196	0	28294
DoS	79	0	0	0	18794	18873
Sum	29320	552	4937	28346	19108	82263

Table F-2: Detailed Accuracy by Class using DT algorithm without feature selection

	TP	FN	FP	TN	TP Rate	FP Rate	precision	Specificity	support
Benign	29108	661	212	52282	0.977795694	0.004039	0.992769	0.995961	29769
Bot	450	100	102	81611	0.818181818	0.001248	0.815217	0.998752	550
Brute	4721	56	216	77270	0.988277161	0.002788	0.956249	0.997212	4777
DDoS	28196	98	150	53819	0.996536368	0.002779	0.994708	0.997221	28294
Dos	18794	79	314	63076	0.995814126	0.004953	0.983567	0.995047	18873
Avg/ Total					0.955321033	0.003161	0.948502	0.996839	82263

Table F-3: Confusion Matrix using NN algorithm without feature selection

	BENIGN	Bot	Brute	DDoS	DoS	Sum
BENIGN	28558	55	674	83	399	29769
Bot	228	244	0	78	0	550
Brute	51	0	4688	2	36	4777
DDoS	114	12	6	28064	98	28294
DoS	104	0	147	111	18511	18873
sum	29055	311	5515	28338	19044	82263

Table F-4: Detailed Accuracy by Class using NN algorithm without feature selection

	TP	FN	FP	TN	DR Rate	FP Rate	Precision	Specificity	support
Benign	28558	1211	497	51997	0.959320098	0.009468	0.982895	0.990532	29769
Bot	244	306	67	81646	0.443636364	0.00082	0.784566	0.99918	550
Brute	4688	89	827	76659	0.98136906	0.010673	0.850045	0.989327	4777
DDoS	28064	230	274	53695	0.991871068	0.005077	0.990331	0.994923	28294
Dos	18511	362	533	62857	0.98081916	0.008408	0.972012	0.991592	18873
Avg /Tot					0.87140315	0.006889	0.91597	0.993111	82263

Appendix G: Sample Source Code

In [16]:
```
from sklearn import tree
d_tree = tree.DecisionTreeClassifier()
d_tree = d_tree.fit(X_train,Y_train)
```

In [17]:
```
from sklearn import tree
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score

kf = KFold(n_splits=10, random_state=None, shuffle=False)
d_tree = tree.DecisionTreeClassifier()
max_accuracy = 0

for train_index, test_index in kf.split(X):
    cur_fold_train_x, cur_fold_test_x = X[train_index],X[test_index]
    cur_fold_train_y, cur_fold_test_y = Y[train_index],Y[test_index]

    cur_tree = d_tree.fit(cur_fold_train_x,cur_fold_train_y)
    cur_prediction = cur_tree.predict(cur_fold_test_x)
    s = accuracy_score(cur_fold_test_y,cur_prediction)
    if s > max_accuracy:
        d_tree = cur_tree
```

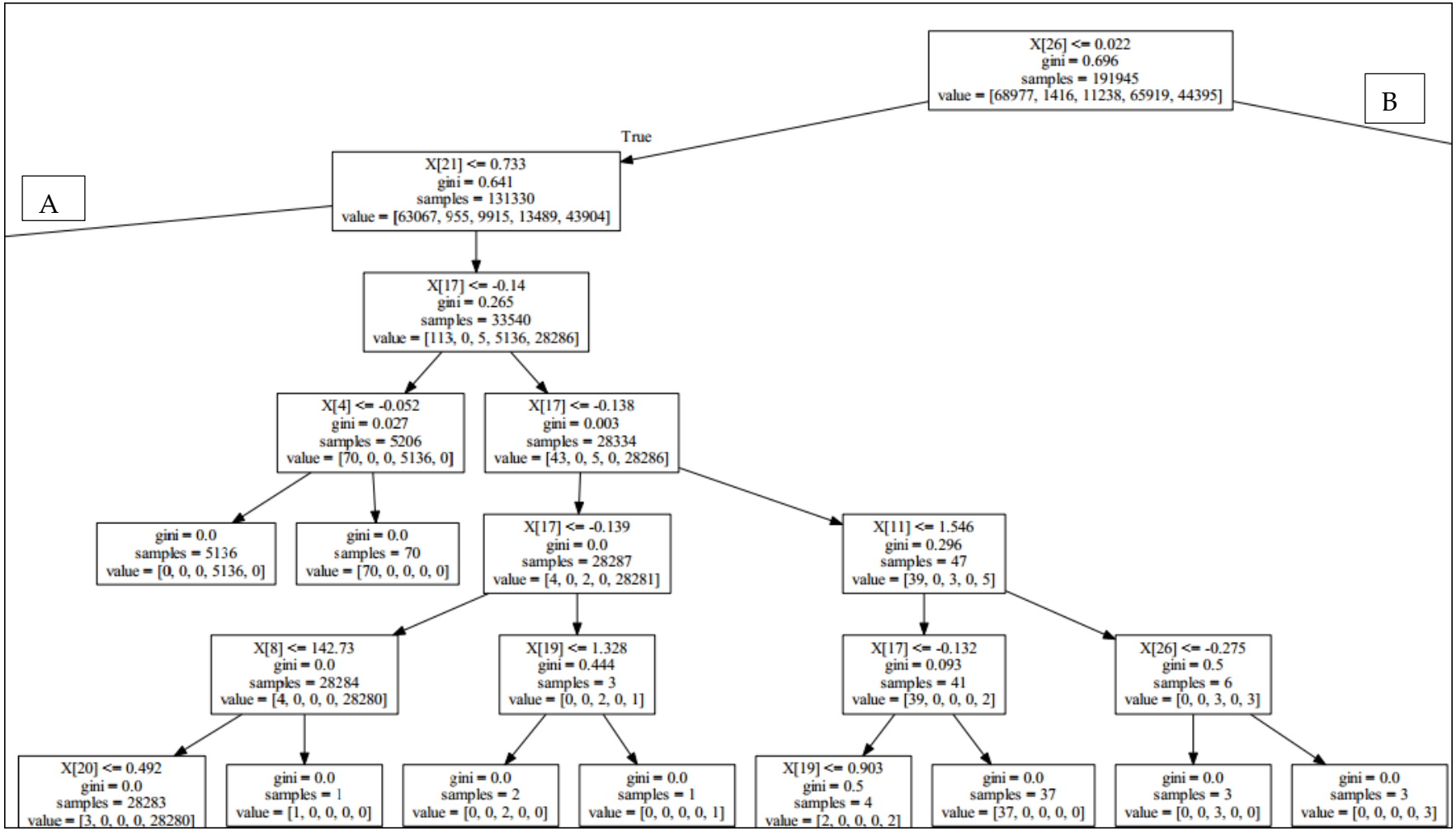
In [19]:
```
from sklearn.metrics import confusion_matrix
predicted_k_means = kmeans.predict(X_test)
x_test[:,:-1] = predicted_k_means
predicted_d_tree = d_tree.predict(X_test)
#+ np.array(predicted_k_means))

confusion_matrix(Y_test,predicted_d_tree, ['BENIGN','Bot','Brute','DDoS','DoS'])
```

Out[19]:
```
array([[29681,    13,     6,    11,    58],
       [    0,   550,     0,     0,     0],
       [    0,     0,  4777,     0,     0],
       [    0,     0,     0, 28294,     0],
       [    0,     0,     0,     0, 18873]], dtype=int64)
```

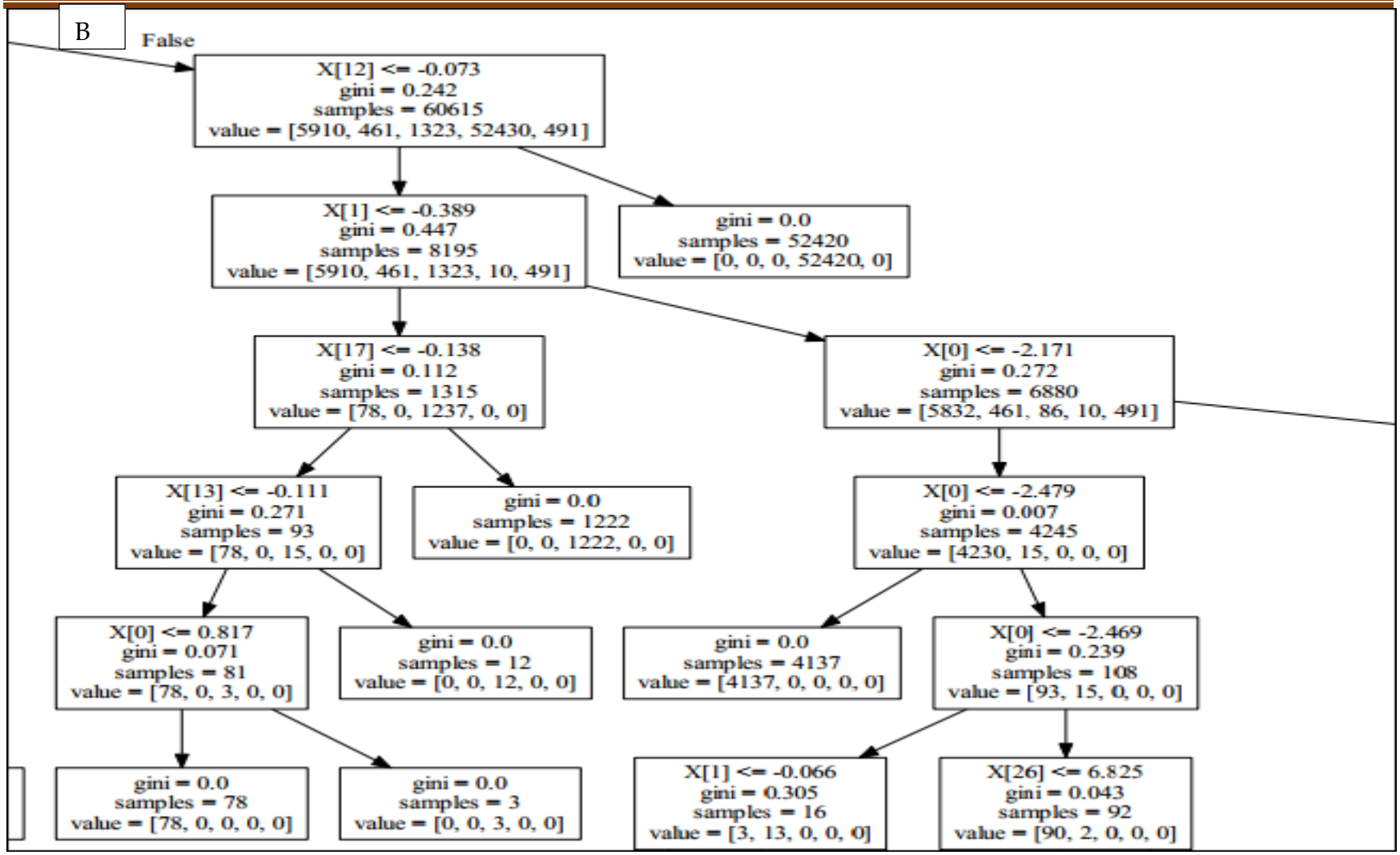
In [22]:
```
from sklearn.metrics import accuracy_score,precision_score
predicted_dtree = d_tree.predict(X_test)
print "Accuracy", accuracy_score(Y_test, predicted_dtree)
print "Precision", precision_score(Y_test, predicted_dtree,average='weighted')
```

```
Accuracy 0.9989302602628156
Precision 0.9989362128429019
```

Appendix H: Sample DT Graph



B


False
X[12] <= -0.073
gini = 0.242
samples = 60615
value = [5910, 461, 1323, 52430, 491]
X[1] <= -0.389
gini = 0.447
samples = 8195
value = [5910, 461, 1323, 10, 491]
gini = 0.0
samples = 52420
value = [0, 0, 0, 52420, 0]
X[17] <= -0.138
gini = 0.112
samples = 1315
value = [78, 0, 1237, 0, 0]
X[0] <= -2.171
gini = 0.272
samples = 6880
value = [5832, 461, 86, 10, 491]
X[13] <= -0.111
gini = 0.271
samples = 93
value = [78, 0, 15, 0, 0]
gini = 0.0
samples = 1222
value = [0, 0, 1222, 0, 0]
X[0] <= -2.479
gini = 0.007
samples = 4245
value = [4230, 15, 0, 0, 0]
X[0] <= 0.817
gini = 0.071
samples = 81
value = [78, 0, 3, 0, 0]
gini = 0.0
samples = 12
value = [0, 0, 12, 0, 0]
gini = 0.0
samples = 4137
value = [4137, 0, 0, 0, 0]
X[0] <= -2.469
gini = 0.239
samples = 108
value = [93, 15, 0, 0, 0]
gini = 0.0
samples = 78
value = [78, 0, 0, 0, 0]
gini = 0.0
samples = 3
value = [0, 0, 3, 0, 0]
X[1] <= -0.066
gini = 0.305
samples = 16
value = [3, 13, 0, 0, 0]
X[26] <= 6.825
gini = 0.043
samples = 92
value = [90, 2, 0, 0, 0]