

# **A Comprehensive Analysis of Intrusion Detection System in Internet of Things**

By

Teh Boon Seong

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMMUNICATION AND NETWORKING

Faculty of Information and Communication Technology

(Kampar Campus)

JANUARY 2020

## REPORT STATUS DECLARATION FORM

**Title:** A COMPREHENSIVE ANALYSIS OF INTRUSION DETECTION  
SYSTEM IN INTERNET OF THINGS

**Academic Session:** JANUARY 2020

I TEH BOON SEONG

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in


Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

**Address:**

7, Jalan Hor Hock Lung

Camay Park,

31650 Ipoh, Perak.

DR. VASAKI A/P PONNUSAMY

Supervisor's name

**Date:** 23 APRIL 2020

**Date:** : 23 APRIL 2020

# **A Comprehensive Analysis of Intrusion Detection System in Internet of Things**

By

Teh Boon Seong

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)


COMMUNICATION AND NETWORKING

Faculty of Information and Communication Technology

(Kampar Campus)

## DECLARATION OF ORIGINALITY

I declare that this report entitled “**A COMPREHENSIVE ANALYSIS OF INTRUSION DETECTION SYSTEM**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :  \_\_\_\_\_

Name : TEH BOON SEONG

Date : 23 APRIL 2020

## **ACKNOWLEDGEMENTS**

I would like to say thank you and appreciation to my dearest supervisors, Dr Vasaki a/p Ponnusamy who gives me an opportunity to take part in cyber security related project. It opens the first door for me in cyber security field which makes me felt interesting in this field. Nevertheless, I would like to thank my family who constantly giving support to me when I faced any sort of problem in doing this research. It is a wonderful experience to took part in this field. Words is not able to fully express my sincere thank you to my supervisor and my family.

### **Abstract**

This project is a project for academic purpose. Methodology, proposed solution, literature review about the types of intrusion detection system (IDS) will be provided to the student. This project will be illustrating the process of training a model to have the capability to detect malicious traffic packet. To train this model, prototyping is used because the model is upgraded or train with more data to increase its accuracy. The process involved in training a machine learning model consist of four step which is data collect the relevant data, data pre-processing, select the feature and classify. The machine learning classification technique used in this project is mainly decision tree, random forest and naïve bayes. Besides, this project allow student to know more about how IDS works in different network and what are the placement strategy. There are 3 types of network will be mentioned in this project which is wired network, wireless network and ad hoc network. In addition, the placement strategy for IDS includes centralized and distributed. Nonetheless, the most interesting part which is the type of IDS includes signature based IDS, anomaly based IDS, host based IDS and network based IDS. This paper also includes the type of data collection in a normal IDS. The main purpose of this project is to increase the accuracy and reduce fake alerts for an IDS.

## TABLE OF CONTENTS

<b>TABLE OF CONTENTS</b>	<b>i</b>
<b>LIST OF FIGURES</b>	<b>iv</b>
Chapter 1: Introduction	1
1.1: Problem Statement	1
1.2: Project Scope	1
1.3: Project Objective	1
1.4: Impact, Significance and Contribution	2
1.5: Background Information	2
Chapter 2: Literature Review	4
2.1 Target Network	4
2.1.1 Wired Network	5
2.1.2 Wireless Network	5
2.1.3 Ad Hoc Network	6
2.2 Types of Intrusion Detection System	8
2.2.1 Signature Based Intrusion Detection System	8
2.2.2 Anomaly Based Intrusion Detection System	9
2.2.3 Network Based Intrusion Detection System	10
2.2.4 Host Based Intrusion Detection System	11
2.3 Deployment of Intrusion Detection System	12
2.3.1 Centralized Intrusion Detection System	13
2.3.2 Distributed Intrusion Detection System	14
2.3.3 Mobile Intrusion Detection System	15

Table of Contents

2.4 Data Collection Method	16
2.4.1 Behaviour Based Collection Method	16
2.4.2 Traffic Based Collection Method	17
Chapter 3: Proposed Method or Approach	33
Chapter 4 Preliminary Result	62
Chapter 5: Discussion	78
Chapter 6: Conclusion	84
<b>Bibliography</b>	<b>85</b>
<b>Appendix</b>	<b>c-1</b>
<b>Poster</b>	
<b>Biweekly Report</b>	
<b>Plagiarism Result</b>	
<b>FYP2 Check List</b>	



**LIST OF FIGURES**

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 2.0	Focus of Intrusion Detection System	4
Figure 2.1	Target Network	4
Figure 2.2	Types of Intrusion Detection System	8
Figure 2.2.1	Aumreesh el al. (2017) describe Misuse/ Signature Based IDS	8
Figure 2.2.2	Aumreesh el al. (2017) describe Anomaly Based IDS	9
Figure 2.2.3	Aumreesh el al. (2017) describe Network IDS	10
Figure 2.2.4	Aumreesh el al. (2017) describe Host Based IDS	11
Figure 2.3	Deployment of IDS	12
Figure 2.3.1	Ghorbani et al. (2009) describes Centralized IDS	13
Figure 2.3.2	Huang et al. (2010) describes Distributed IDS	14
Figure 2.3.3(a)	Gao & Jin (2010) describes Mobile Agent	15
Figure 2.3.3(b)	Gao & Jin (2010) describes Lifecycle of Mobile Agent	15
Figure 2.4	Data Collection Method	16
Figure 2.5 (a)	Diagram of the flow of the classification machine learning	19
Figure 2.5(b):	L. Dhanabal & Dr. S.P. Shantharajah (2015) describes attribute value type	20
Figure 2.5(c)	L. Dhanabal & Dr. S.P. Shantharajah (2015) describe details of normal and attack data in different type of NSL-KDD data set	20

## List of Figures

Figure 2.5(d)	L. Dhanabal & Dr. S.P. Shantharajah (2015) describe mapping of attack class with attack type	21
Figure 2.5(e)	Data set before format into .csv	20
Figure 2.5(f)	Data set after format into .csv	21
Figure 2.5(g)	Attack types in words	21
Figure 2.5(h)	Attack types further label in number	21
Figure 2.5(i)	Attack types label according to the number	22
Figure 2.5(j)	Machine Learning Model	24
Figure 2.5(k)	Sample wireshark packet	27
Figure 2.5(l)	Detail of packet	28
Figure 2.5(m)	Detail of ICMP packet	28
Figure 2.5(n)	Normal traffic	29
Figure 2.5(o)	Result and graph of type of machine learning classification techniques and different test size	30
Figure 3(a)	Work Flow of This Research	31
Figure 4(a)	Work Flow of This Research	62
Figure 4(b)	Top 10 feature scores for NSL-KDD dataset	64
Figure 4(c)	Aumreesh el al. (2017) describe Anomaly Based IDS	64
Figure 4(d)	Results for Decision Tree Model(NSL-KDD)	67
Figure 4(e)	Results for Naïve Bayes Model(NSL-KDD)	69

## List of Figures

Figure 4(f)	Results for Random Forest Model(NSL-KDD)	70
Figure 4(g)	Results for Support Vector Machine Model (NSL-KDD)	72
Figure 4(h)	Results for Decision Tree Model(NaBIoT)	73
Figure 4(i)	Results for Naïve Bayes Classifier (NaBIoT)	74
Figure 4(j)	Results for Random Forest Model (NaBIoT)	76
Figure 4(k)	Results for Support Vector Machine Model (NaBIoT)	77
Figure 5(a)	Accuracy of Different Combination of Features (NSL- KDD)	79
Figure 5(b)	Accuracy of Different Combination of Features (NaBIoT)	80
Figure 5(c)	False Positive Rate for Different Combination of Features(NSL-KDD)	81
Figure 5(d)	True Positive Rate for Different Combination of Features(NSL-KDD)	81
Figure 5(e)	False Positive Rate for Different Combination of Features(NaBIoT)	82
Figure 5(f)	True Positive Rate for Different Combination of Features (NaBIoT)	82
Figure 5(g)	Comparison of NSL-KDD and NaBIoT	83

## Chapter 1: Introduction

### 1.1 Problem Statement

In wireless network, it always exposed to a lot of security issue such as Denial of Service(DoS) attack, wireless hijacking, authentication attack and other type of attacks that are targeted in wireless network. Therefore, an intrusion detection system (IDS) works as an alarm mechanism for computer system. It detects any malicious activity happened to the computer system and it alerts an alarm message to notify user there are malicious activity. There are IDS that are able to take action when malicious or anomalous network was detected, which include suspend the traffic sent from suspicious IP address. There are only few comprehensive work that cover some of the IDS and did not give a concluding remark on what is the advantage and disadvantage of the IDS in different type of network and placement. Therefore, what is the advantage or disadvantage of different type of IDS, placement strategy data collection method and the types of network that the IDS should be place. A paper done by J. Amudhavel et. al. 2016 covers the challenges of the different type of IDS but did not emphasize the advantage of the IDS. Another research done by P. Sadotra and Dr. C. Sharma only listed out the type of IDS, placement strategy, reaction on intrusion and targets analysis timing but it did not have the advantage or disadvantage of the IDS.

### 1.2 Project Scope

The project scope for this research is to review the type of IDS, placement strategy and how the data are collected. Besides, the classification technique is going to be tested to find out the suitable algorithm in IDS. Besides, the other concern is to reduce the false positive rate and false negative rate. The proposed solution is using machine detect they different type of attack may happen within a network. In this research, python code is used to train the model in order to detect the types of traffic and determine whether it is a malicious traffic or normal traffic.

### 1.3 Project Objectives

Main objective for this research is to proposed a machine learning IDS which are capable to predict or detect any malicious traffic. To train a machine learning model, different type of malicious traffic which consist of different type of attacks was used to

be the training data for the machine learning. By using different machine learning classification algorithm to determine the best fir algorithm in IDS. This project will mainly focus on the accuracy of detection of IDS. To achieve the objective, we have sub-objectives that act as a milestone for reach the main objective.

The first sub-objective is to analyse different type of IDS and the placement strategy of IDS such as distributed or centralized. It enables us to understand more how a IDS works in different type of network such as wired, wireless and ad hoc network. The second sub-objective for this research is to determine the best suit machine learning classification algorithm in detecting malicious traffic. Every algorithm works different when comes to machine learning, the best algorithm should be chosen as the one due to IDS is a crucial part before enter to a network because it will be able to detect most of the traffic after it is unable to filter out by firewall. The third sub-objective is to determine the type of dataset and machine learning algorithm that gives the best result in detecting anomaly and normal traffic.

#### 1.4 Impact, Significance and Contribution

With the implementation of machine leaning to predict or detect whether it is a malicious traffic or normal traffic, the detection accuracy is able to be increased. This is when a model has been train by different type of attack or malicious packet. It is able to predict whether the next packet coming from the traffic is a malicious or normal. The proposed method is trying to minimize the detection error in IDS because if there are any malicious traffic entered to an organization or government network. The damage dealt by the attack is huge or may cause hundreds of millions in recovering the data or network. in this project it allows the end user or any administrator to have a IDS with the lowest error.

#### 1.5 Background and motivation

IDS is a system which will monitor the network traffic and alert a notification to the administrator if there is any suspicious or malicious activity. Besides intrusion detection system, there is another system which will have the same role as intrusion detection system but with additional feature which will reject or drop the packet if the network traffic is malicious. But in this project we are going to focus on intrusion detection system only. To determine the accuracy of an intrusion detection system, there are 4 states which is true positive, true negative, false positive and false negative.

For true positive, this is the state where a malicious packet which contain any sort of attacks in coming into a network and the intrusion detection system is able to detect it and raise an alarm to the administrator. The second state which is true negative, this is a state which does not detect any attacks and the traffic is normal. The third state is false positive, it will trigger the alarm and notify the administrator there is an intrusion but in fact the network traffic is normal. In the final state which is the most dangerous which is false negative, the network traffic is malicious and the intrusion detection system did not raise any alarm to notify the administrator regarding the malicious traffic which may cause damage to the network.

### 1.6 Proposed study

Therefore, this project is using machine learning to create an intrusion detection system. The goal for machine learning is to have a model which is able to predict the incoming network traffic whether it is malicious or normal. In this project there are a few classifiers used to develop the machine learning such as decision tree classifier, random forest, naïve bayes and support vector machine. These classifiers consist of different kinds of algorithm to predict the network traffic. In addition, the NSL-KDD dataset is compared with wireless dataset to identify the difference between feature selection of this two dataset.

The achievement from the previous work is a model is trained with NSL-KDD dataset was successfully build. The results from different train size for the respective classifier is recorded. The most accurate classifier achieve from the previous work is decision tree classifier and the least accurate is random forest.

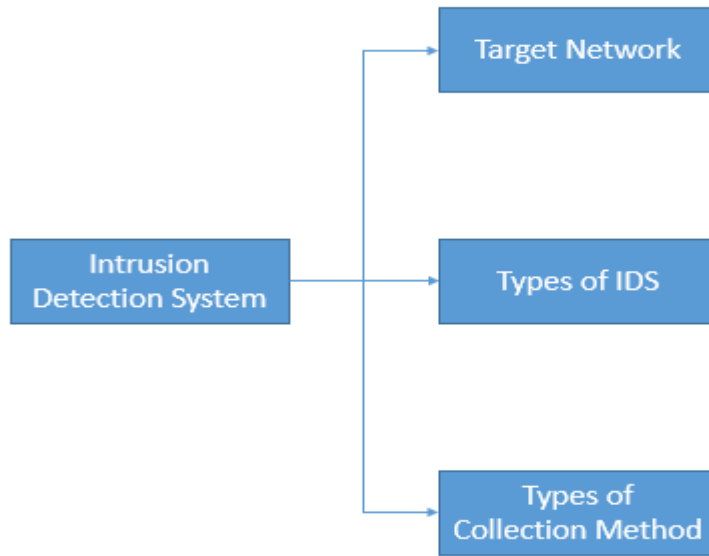


Figure 2.0 Focus of Intrusion Detection System

### 2.1 Target Network

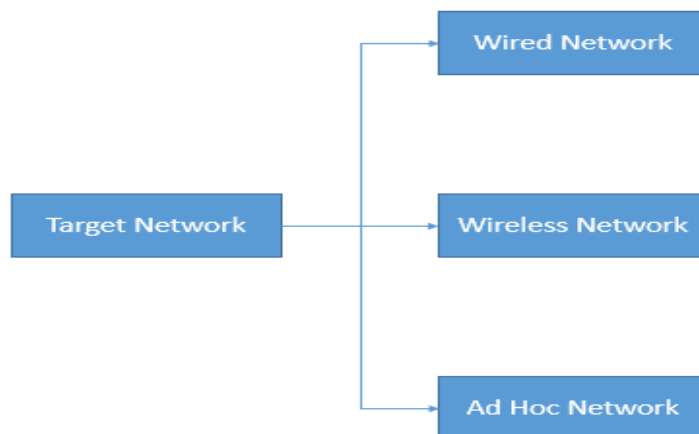


Figure 2.1 Target Network

### **2.1.1 Wired network**

A group of computer or device connected via network links are wired network. the objective is to transmit data between computers or device using optical cables or transmission medium or to share resources. Wired network did not expose to attack as much as wireless network because to access the data transmitted between two computers in wired network needs access to one of the computer or network jack or cable. Therefore, wired network are more secure compare to wireless network. But there are not 100 percent guarantee that it is not vulnerable to attacks, in wired network it has vulnerability too such as misfeasors who are legitimate accessing to the data but he or she does not grant an access for it. It also has advantages in speed and the cost usually decide by the element of the network such as number of computer or the amount of cable needed. Consequently, it makes this network cheaper and more affordable rather than using wireless network and wireless network may cause signal loss or fading due to interference. (Radja 2015) The disadvantage of wired network is if there is any wired network failed or destroyed in, it will cause the whole system completely immobilized. For example, if one of the wired network is failed it will affect the safety of the production of coal mine. (Zhang & Mao 2015)

### **2.1.2 Wireless Network**

In this modern era, wireless networks had become an important part of the connectivity operation of smartphone. There are various kind of wireless network such Wi-Fi, cellular and Bluetooth. Social media applications, online chatting apps, video streaming apps and etc. uses wireless network to enable the transmission of data. Out of all the wireless network, Wi-Fi and cellular network are the most predominant due to its advantageous feature such as ubiquitous access, availability and tolerable budget. (Rattagan 2016) The availability of wireless Local Area Network gives an advantage in the application in the market. Smartphone user is able to connect Wi-Fi wireless internet “hotspot” connection in public, therefore it makes the Wi-Fi vulnerable to attacks or intrusion. This vulnerability can do harm to the user because the hacker is able to commit fraud, steals personal information, identity theft and more. Rouge access point can be set by attacker to misdirect the user that it is a legitimate Internet access point, but actually it is used to eavesdrop the wireless communication among Internet surfers. (Vanjale & Mane & V.Patil 2015) The number of attacks increase exponentially



due to Wi-Fi network is widely used for high-speed local area connectivity. The security measure used in Wi-Fi is IDS which is a detection system widely used for every network security infrastructure. (Aminanto et al. 2017) When an attacker had sufficient traffic it can be easily cracked a Wi-Fi due to its weaknesses at cryptographic. On the data link layer, it is still vulnerable to attacks because the protocol design must not be encrypted. Therefore, the network is still at risk for Denial of Service(DOS) attack and which are resulted from packet forgery. Companies and institution had set up distributed Wi-Fi networks for better network access. The distributed network uses pre-authentication to the access point. With this implementation it allows the user to connect to the access point in a larger area and stay connected no matter where he/she moves the facility by using different access point. But then again, it is still at risk to denial of service attacks and packet forgery. The distributed network allows the administrator to identify if there is any attacks on a particular location, it can be detects the location of the attacker to prevent any further damage to the network. (Satam 2017)

### **2.1.3 Ad Hoc Network**

In a network that contain mobile nodes which is able to communicate with one another without any exact infrastructure is ad hoc network. Mobile ad hoc network is a group of self-sufficient mobile nodes which is able to talk to each other via wireless links. There is a problem in ad hoc network which is it has a limitation in wireless range as every host needs the help of nearby hosts to forward the packet from source to destination. When comes to the sensitivity to detects attacks, Mobile Ad-Hoc Network(MANET) has an advantage than the wired network infrastructure since it contains a very limited physical protection constrain. Reputation management is more appropriate than traditional intrusion detection which are claimed by some researchers. Ad hoc network IDSs concentrates in distributed design, due to highly transient population differentiate ad hoc network from other wireless application. One of the method that is suitable to deploy IDS in ad hoc network is signature based intrusion detection. Signature based intrusion detection will be looking for the runtime that match a specific pattern of misbehaviour. This category consists of low false positive rate which is a major advantage for this approach. With this approaches, it only reacts to known bad behaviour. A normal node will not show the attack signature by theoretical basis. On the other hand, the key disadvantage for this approach is that it's a technique that will look for a specific pattern. To be able to search any detailed pattern, a

dictionary must state each attack vector and stay current. The main focus of problem in this field is to create an accurate attack dictionary. Signature length is an indicator of efficiency for the signature based intrusion detection, on the other hand signature which is longer than the usual one will have a bigger memory requirement and more powerful microprocessor use to detect it. This approach is effective to outsider attacks; malicious outsiders will have a well-known signature in the course of penetrating the network. (R. Mitchell & C. Ing-Ray 2014) There are 2 different type of architecture of ad hoc network, one of it is isolated. In the same network, the nodes that are able to communicate between them is an isolated ad hoc network. In isolated ad hoc network, it can be classified into 2 type which is large scale and small scale isolated ad hoc network. There may be thousands of nodes in a large scale isolated network. This network is not suitable for transmitting large amount of data due to security problems. One of the weakness of this network is it will expose to higher security problem. Besides, the architecture cost of this network is relatively high and the traffic performance is also low. Small scale isolated ad hoc network able to raise the uses in commercial such as smart home, business meeting place, hotspots and some private area. The last type of ad hoc network is integrated; it can be seen in the following scenarios. The first scenario can be hotspot, in this modern era, Smartphone is able to become a hotspot that is highly secure or can be the source of internet for phone, PC and other devices. The other scenario can be GPRS, any member of ad hoc network can access to Internet by using GPRS. This scenario has a disadvantage in data rate, it has restriction to it as compare Hotspot. On the other hand, this network gives user benefits in some cases such as in airport or railway station. (Sharmilla. S & Shanthi. T 2016)

## 2.2 Types of Intrusion Detection System

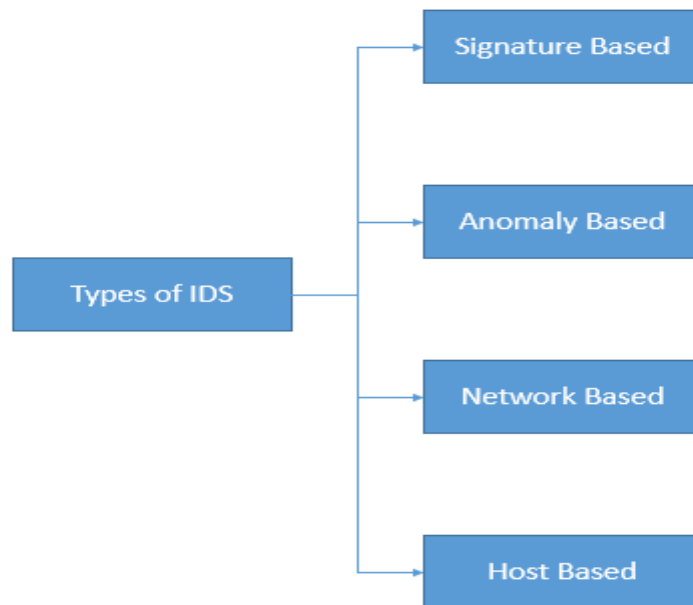


Figure 2.2: Types of Intrusion Detection System

### 2.2.1 Signature-Based Intrusion Detection System

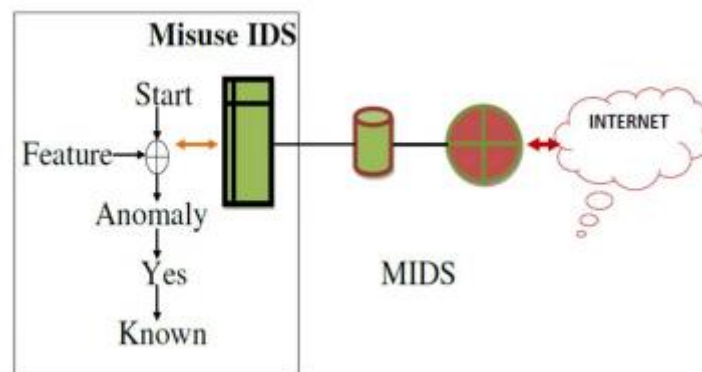


Figure 2.2.1(a) Aumreesh et al. (2017) describe Misuse/ Signature Based IDS

Signature based IDS detects intrusion based on the pattern matching mechanism to the IDS database with the signature of the attacks. If the traffic contains one or more of the signature pattern in the IDS database, the IDS will detect it and identify it as a malicious traffic or attack. Comparing tools will verify all of the data by comparing it to the IDS database. If the packet did not exhibit any malicious pattern it will be send to the destination network. The probability for a signature based mechanism to generate fake alert is relatively low. (B.I. Santoso et. al. 2016) Because

of the signature mechanism, the false positive rate for this IDS is low. This approach can be a great detection method when comes to known pattern. This is because a good node will not have the pattern of the attack signature. In addition, signature based IDS gives a better protection against outsider attacks because malicious outsider usually has a specific malicious pattern in when they are trying to attack the network. Besides the advantage, there are also have a downside to it. Signature based technique must look for similar pattern in order to effectively detect any malicious packet. Therefore, the IDS database or dictionary must be constantly update to detect any new malicious pattern. If the dictionary did not stay up to date, the detection of new malicious pattern will be considered as normal traffic. (R. Mitchell & C. Ing-Ray 2014)

### 2.2.2 Anomaly-Based Intrusion Detection System

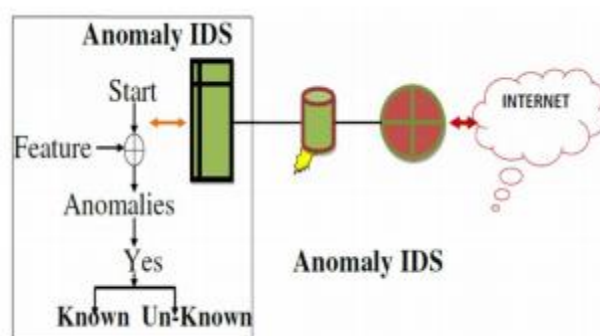


Figure 2.2.2(a) Aumreesh et al. (2017) describe Anomaly Based IDS

Anomaly based IDS will detect any intrusion by identifying any difference in behaviour of an ordinary traffic. For instance, if a normal traffic is a usual day suddenly acts differently it may indicate that the computer is being attacked and the data is redirected to the attacker. (N.T Van & T.N Think & L.T. Sach 2017) There is various anomaly detection can use data mining techniques. readymade data mining techniques that can be applied directly to detect intrusion. There are 4 classes of data mining which is they are association rule learning, clustering, classification and regression. Clustering based anomaly detection techniques, the data that divided into group of object with the same characteristics is clustering. Cluster consists of object that are alike in the same group but different in the other group. Clustering algorithms can detect any intrusion without prior knowledge. Classification based anomaly detection, classification is classifying the category of new instances on the basis of a training set of data containing instances whose category membership is known. Classification in machines learning

can be considered as an instance of supervised learning like learning a training set of correctly-identified observation is available. Algorithm that implements classification is classifier. Hybrid approach is merging different algorithms together to have a better detection where using any particular algorithm is not sufficient to yield proper result. A technique to train in supervised mode is supervised anomaly detection, which predict the accessibility of a training data set that labelled as normal or anomaly class. This approach is usually to construct a predictive model for normal versus anomaly classes. (Amanpreet & Mishra & Kumar 2012) There are two problems which arise in supervised anomaly detection. The first problem is normal instances in the training data is greater than anomaly instances. This is due to the class distribution is not balanced that were addressed in the data mining and machine learning literature. Accurate and representative labels especially for anomaly class is quite challenging which is the second problem. Semi-Supervised anomaly detection is a technique which operate in semi-supervised mode and only normal class is available in the training data. Semi-supervised anomaly detection labels for anomaly class is not require, which makes it more applicable as compare to supervised techniques. Usually this approach will build a model for the class corresponding to normal behaviours and the model was used to determine the anomalies in the test data. Unsupervised anomaly detection uses a technique that runs in unsupervised mode and training data is not required therefore it is applicable in most situation. This detection makes implicit assumption that anomalies is less frequent than normal instances in test data. The rate of false alarm is high if the assumption is incorrect. (R. Mitchell & C. Ing-Ray 2014)

### **2.2.3 Network-based Intrusion Detection System**

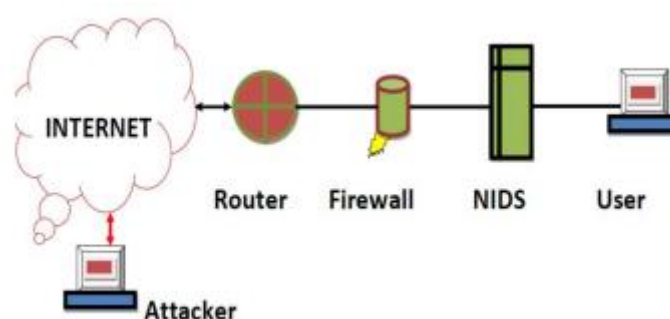


Figure 2.2.3(a) Aumreesh el al. (2017) describe Network IDS

Detection tools that implement network based approach towards an intrusion is a network-based IDS. The whole network's traffic from which the host are connected will be monitor by network-based IDS. Network-based IDS is able to gives real time detection of the network attacks then it can reduce or decreases the chances of the damage of the network dealt by the intrusion and it is cost effective. (Muhammad K. Asif et al.2013) In addition, computer on the same network can be protected by other IDS but for network-based IDS it can only the computer located on the same network and the information about routing from different system to the IDS. On the other hand, network-based IDS has a major drawback which is if the packet is encrypted with any sort of encryption algorithm, network-based IDS is unable to read content of packet. (A.T Taha et. al. 2015) Besides there are another advantage of network-based IDS, it is a system that has preferred standpoint of organizing and observe for attacks, that are able to be establish on the entire system. Constantly observe for attack allows the system to provide an excellent detection. (Aumreesh el al. 2017)

#### **2.2.4 Host-based Intrusion Detection System**

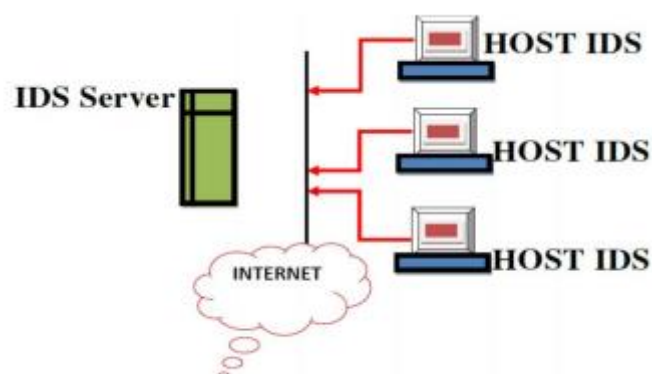


Figure 2.2.4(a) Aumreesh el al. (2017) describe Host Based IDS

Host based IDS is a type of IDS that will allocate on a specific host on the network. Host-based approach is mainly aimed to protect a single computer, and that single system was prevented to execute malicious code. The selection of metric is done by host-based IDS and the decision engine will need the metric provided as an input. The metric feature is needed to offer to the set of methods that involved in the data collected from different log files that occur in the system. (Sandeep & Thaksen 2016) If there are any attribute value of a new record is above the threshold that is measured by the system, the system will generate alerts. To identify anomalies, it can be done by

using multivariate statistical analysis on audit records. Besides, in shell commands logs, it can be detecting through frequency distribution based anomaly detection. In addition, there are another type of host-based IDS that will train a different algorithm on system call of normal software behaviour. If the unknown software behaviour is observed with normal system calls, an alert will be raised due to anomalies was detected. (Murtaza et al. 2013) Host-based IDS brings a major advantage in analysing the attack that are successful intrude into the network. Caution can be created by the system subordinate framework about any action that are closely related to attacks. If any arranged activity is jumbled up, then the entrance of movement in a decoded shape that are done by the host-based observing framework. (Aumreesh et al. 2017) There are a few weakness possess on the use of log files. The first problem which is also the most basically, interpreted data is represented as log files. Daemon programs monitoring system activity will produce log files and inherently and irrevocably deliver a thinned data sources. The last problem which is the effectiveness of disk zones. The creation of log files will produce a huge number of potential irrelevant data which is almost equivalent priority to critical data. Besides, it also goes along with mechanical problem such as managing and creating log files. (Creech & Hu 2014)

### **2.3 Deployment of Intrusion Detection System**

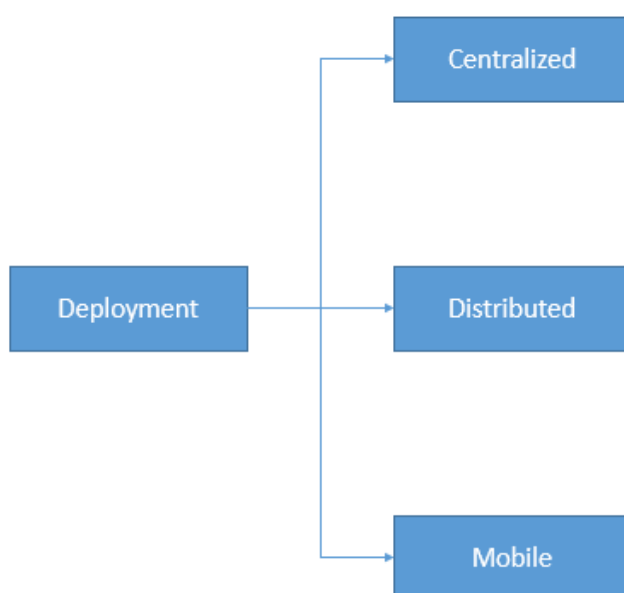


Figure 2.3 Deployment of IDS

### 2.3.1 Centralized Intrusion Detection System

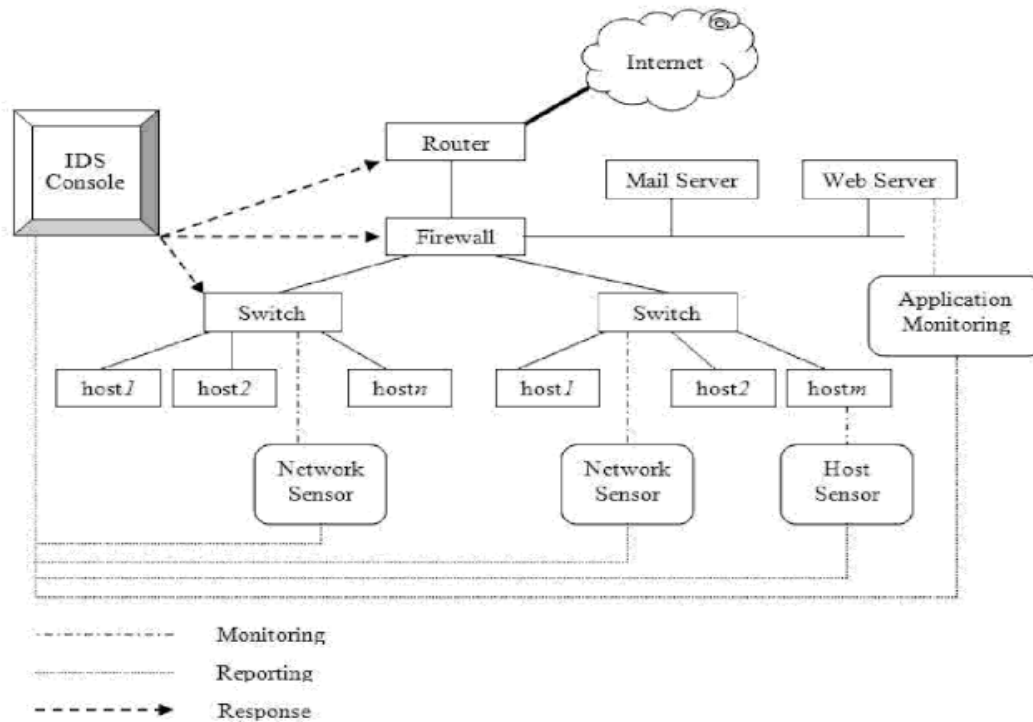


Figure 2.3.1 Ghorbani et al. (2009) describes Centralized IDS

Centralized IDS, the analysis of packet will be done in one or a small number of nodes. Audit component of centralized IDS will be distributed, but the collected audits will be traverse to a particular place for analysis to take place. (Toulouse & Minh & Curtis 2015) Centralized IDS will generate different types of alerts and agents will analyse the network node or host. The alert will be transfer to a central C&C handler which are responsible for analysing and making an accurate decision. (L.N. Tidjon & M. Frappier & A. Mammam 2019)



### 2.3.2 Distributed Intrusion Detection System

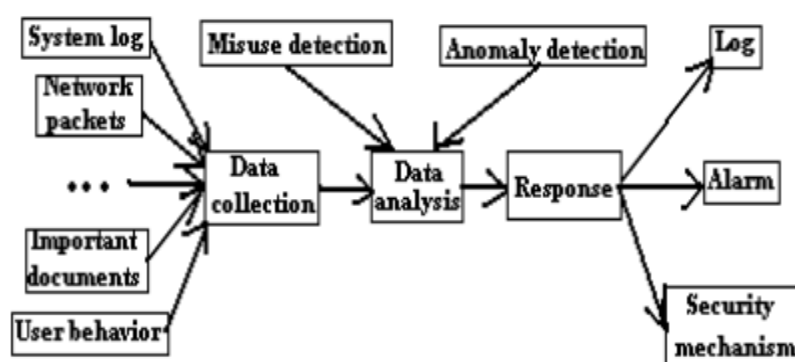


Figure 2.3.2 Huang et al. (2010) describes Distributed IDS

Grouping of many intrusion detection systems over a large set of network can establish distributed IDS. To establish communication between a single server with multiple clients, centralization method of communication was applied. (J. Amudhavel et al. 2016) distributed IDS able to let the infrastructure to detect a coordinated attack to an organization and any distributed resources that are related to the organization. (Vandana P et. al. 2014) A IDS which have the combination of network based IDS or host based IDS is distributed IDS. The basic component should be included in the IDS is detection mechanism and correlation manager. Detection mechanism usually check or monitor the entire network traffic and transfer the information gathered to correlation manager. The task for correlation manager is to perform global correlation of information from different IDS and generate alert if it is an attack. Therefore, distributed IDS is useful in cloud computing because it has global correlation to detect distributed intrusion and also detect individual intrusion through detection mechanism. (Y. Mehmood et. al. 2015)

### 2.3.3 Mobile Agent Intrusion Detection System

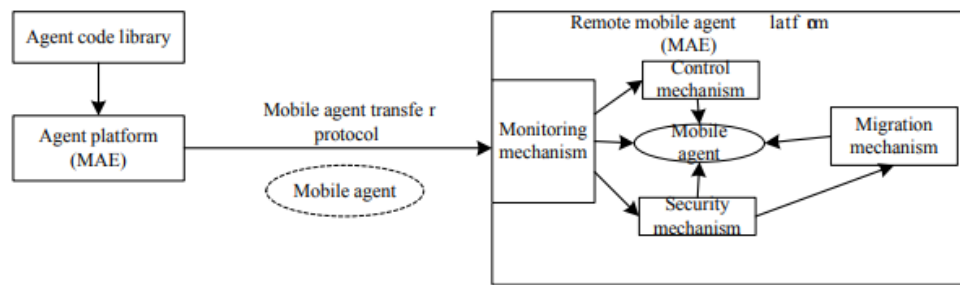


Figure 2.3.3 (a) Gao & Jin (2010) describes Mobile Agent

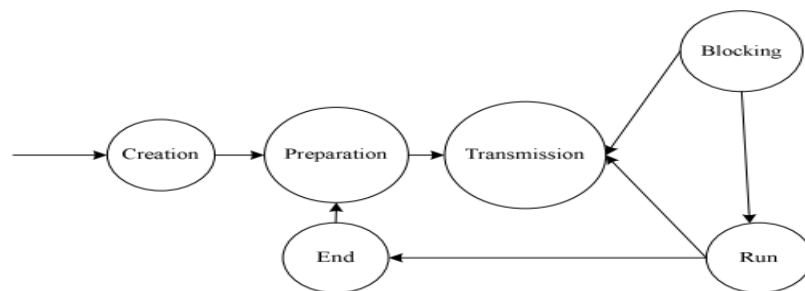


Figure 2.3.3 (b) Gao & Jin (2010) describes Lifecycle of Mobile Agent

An automated entity that able to do different job in order to reach some goal is called mobile agent. Within the domain of networking, an agent is able to operate although the user is disconnected from the network. mobile agent is a software that are able to move around the network and it will complete the goals that set by user. What differentiate an agent between application is the agent are will complete their goal. This is because the agent usually will automatically complete their task that set by the user. Therefore, the agent has the capabilities to control themselves in making any decision that when and where they should be moving. (Y. EL. Mourabit et. al. 2014) There are a few advantage that are given by mobile agent. One of is the network load can be reduced by having the processing algorithm which is an agent to the data rather than sending all of the data to the data pre-processing unit. Besides, network latency also can be overcome. If the agent operates directly on the host, the respond is much quicker compare to a tree based system that needs communication to the central coordinator that is not in the network. In addition, the agent has the privilege to move in different environment and insert an operating system independent layer. It has dynamic adoption as the mobility of an agent can be reconfigured. Special agent will be deployed to the attack's location to collect data from it. The last advantage offers by an mobile agent is

scalability. The computation load of will be divided to different machine and the network load will be reduced if the central processing unit has been replaced by a distributed mobile agent. (Yousef EL Mourabit et. al. 2014)

## **2.4 Data Collection Method**

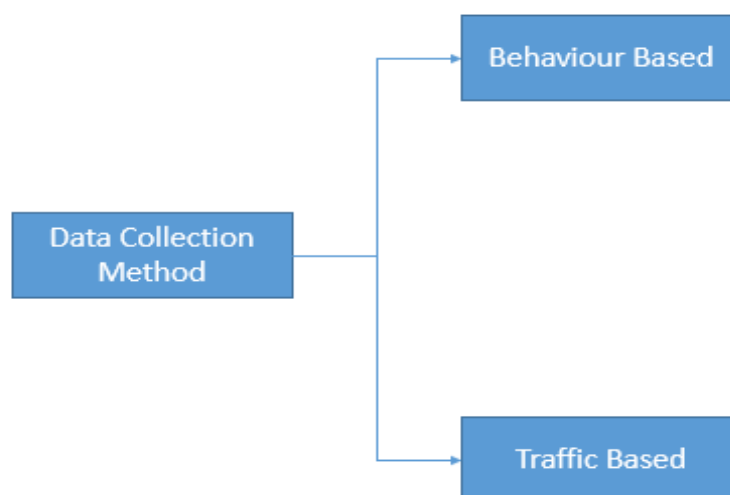


Figure 2.4: Data Collection Method

### **2.4.1 Behaviour Based Collection Method**

One of the collection method is behaviour based collection, to determine whether it is compromised, behaviour based collection will analyse the logs maintained by a node or other audit data. Scalability can be one of the major benefits of behaviour based collection approaches. Large scale network like wireless sensor network and mobile telephony is suitable to use behaviour based collection method. Besides, it is also decentralization which means that it is useful for application like ad hoc network due to its infrastructure-less. No matter how perfect it is, it always has some flaws in it. Behaviour based collection method needs to perform extra effort to collect data, which will increase the workload of the intrusion detection system. it does not have the effectiveness as compare to traffic based collection method. (Mitchell & Chen 2014)

### **2.4.2 Traffic Based Collection Method**

There is another way to collect data which is traffic based collection. To identify whether a node is infected, traffic based collection method will study the network activity. The inspection is either general or protocol-specific. Traffic based collection method has a benefit in resource management, to maintain or analyse their log and individual nodes are free of the requirement. The disadvantage of traffic data collection method is the transparency for collecting audit data from the nodes became a limitation to it. In most of the wireless system, traffic based collection method is better than behaviour based collection method. (Mitchell & Chen 2014)

### **2.5 Preliminary Work**

Machine learning is one of the method and technology that needed to be performed in various setting like the types of IDS, what is the level of the intrusion detection system, placement of IDS, types of data and performance metrics. Machine learning allows the machine to recognize pattern or learn from the data being input into it and the computer was programmed by the human. The conversion of information to knowledge is known as the concept of learning. (Freeman et al. 2017) A machine learning algorithm is “training data, representing, representing experience, and the output is some expertise, which usually takes the form of another computer program that can perform some task.” (Shalev-Shwartz and Ben-David, 2014)

In machine learning there should be 3 types phases instead of just 2, this 3 phases includes training, validation and testing. Machine learning includes 3 approaches which is unsupervised learning, semi-supervised learning and supervised learning. For unsupervised learning, the pattern, structures or knowledge was identified in unlabelled data. Semi-supervised learning, a part of the data was labelled during the gathering of the data or by human experts. This approach greatly helps to solve the problem due to the addition label. For the last one which is supervised learning, the data will be completely labelled. It will find a function or model that able to explain the data. It is useful when comes to model the data to the underlying problem. (Buczak & Guven 2016) Computer is needed to execute the python code for the machine learning part. Besides the hardware needed for this research, software like Spyder is also needed to compile the python code and train a model for the prediction of the packets. On the other hand, traffic's packets can be capture through Wireshark or other software.

Algorithm for the machine learning should be chosen wisely in order to maximize the performance of the machine learning mode. To develop this IDS, Wireshark is used to capture the normal wireless traffic packet and it does not contain any malicious traffic packet. After the normal wireless traffic packet was captured, attacks such as Denial of Service attack, man in the middle attack, ARP spoofing and other wireless attack was used to attack the network and the malicious traffic packet was captured. The normal and malicious wireless traffic packet is used to train the machine learning model and test the model. After the machine learning was trained by using the malicious and normal traffic packet, a decision making was used to determine the placement strategy of the IDS whether signature based IDS, anomaly based IDS, hybrid IDS, network based IDS or host based IDS. This decision making approach is basically based on the types of network that are chosen to deploy the IDS.

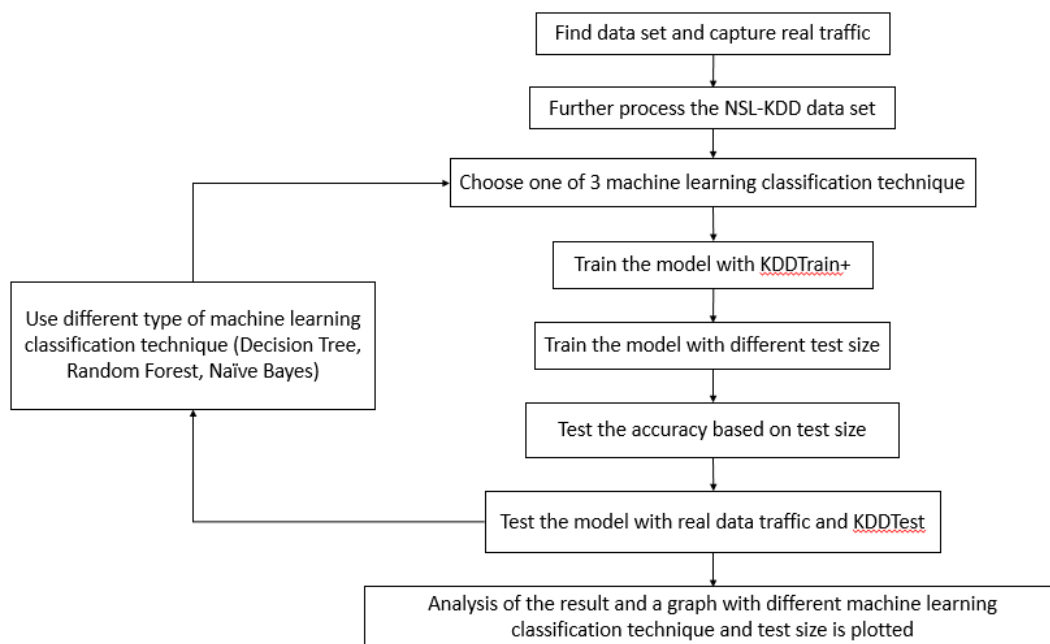


Figure 2.5(a): Diagram of the flow of the classification machine learning

In figure 2.5(a) it shows the work flow of developing the system. First, find data set and capture real traffic. This is the first stage of the development as the malicious and normal traffic data set is found or capture. The data set used to train the machine learning model is NSL-KDD.

Type	Features
Nominal	Protocol_type(2), Service(3), Flag(4)
Binary	Land(7), logged_in(12), root_shell(14), su_attempted(15), is host login(21), is guest login(22)
umeric	Duration(1), src_bytes(5), dst_bytes(6), wrong_fragment(8), urgent(9), hot(10), num_failed_logins(11), num_compromised(13), num_root(16), num_file_creations(17), num_shells(18), num_access_files(19), num_outbound_cmds(20), count(23) srv_count(24), serror_rate(25), srv_serror_rate(26), rerror_rate(27), srv_rerror_rate(28), same_srv_rate(29) diff_srv_rate(30), srv_diff_host_rate(31), dst_host_count(32), dst_host_srv_count(33), dst_host_same_srv_rate(34), dst_host_diff_srv_rate(35), dst_host_same_src_port_rate(36), dst_host_srv_diff_host_rate(37), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41)

Figure 2.5(b): L. Dhanabal & Dr. S.P. Shantharajah (2015) describes attribute value type

Data Set Type	Total No. of					
	Reco rds	Norm al Class	Do S Cla sses	Probe Class	U2R Class	R2L Class
KDD Train+ 20%	25192	13449	9234	2289	11	209
		53.39%	36.65%	9.09%	0.04%	0.83%
KDD Train+	125973	67343	45927	11656	52	995
		53.46%	36.46%	9.25%	0.04%	0.79%
KDD Test+	22544	9711	7458	2421	200	2754
		43.08%	33.08%	10.74%	0.89%	12.22%

Figure 2.5(c): L. Dhanabal & Dr. S.P. Shantharajah (2015) describe details of normal and attack data in different type of NSL-KDD data set



0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	tcp	http	SF	232	8153	0	0	0	0	0	1	0	0	0	0	0	0
0	tcp	http	SF	199	420	0	0	0	0	0	1	0	0	0	0	0	0
0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	tcp	remote_jol	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	tcp	http	SF	287	2251	0	0	0	0	0	1	0	0	0	0	0	0
0	tcp	ftp_data	SF	334	0	0	0	0	0	0	1	0	0	0	0	0	0
0	tcp	name	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	tcp	netbios_ns	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2.5 (f): Data set after format into .csv

After the data set has been format into .csv file, the attack types of the data set which show in figure 3(d) has been further label into number which show in figure 2.5 (e).

0	0	0.05	0	normal
0	0	0	0	normal
1	1	0	0	neptune
0.03	0.01	0	0.01	normal
0	0	0	0	normal
0	0	1	1	neptune
1	1	0	0	neptune
1	1	0	0	neptune
1	1	0	0	neptune
1	1	0	0	neptune
0	0	1	1	neptune
1	1	0	0	neptune
0	0	0	0	normal
0	0	0	0	warezclien
1	1	0	0	neptune
1	1	0	0	neptune

Figure 2.5 (g): Attack types in words

0	0	0.05	0	0
0	0	0	0	0
1	1	0	0	10
0.03	0.01	0	0.01	0
0	0	0	0	0
0	0	1	1	10
1	1	0	0	10
1	1	0	0	10
1	1	0	0	10
1	1	0	0	10
0	0	1	1	10
1	1	0	0	10
0	0	0	0	0
0	0	0	0	21
1	1	0	0	10
1	1	0	0	10

Figure 2.5 (h): Attack types further label in number



0	back	dos	12	phf	r2l
1	buffer_ove	u2r	13	pod	dos
2	ftp_write	r2l	14	portsweep	probe
3	guess_pas	r2l	15	rootkit	u2r
4	imap	r2l	16	satan	probe
5	ipsweep	probe	17	smurf	dos
6	land	dos	18	spy	r2l
7	loadmodul	u2r	19	teardrop	dos
8	multihop	r2l	20	warezclien	r2l
9	neptune	dos	21	warezmast	r2l
10	nmap	probe	22	normal	normal
11	perl	u2r	23	unknown	unknown

Figure 2.5(i): Attack types label according to the number

In figure 2.5f) the attack type in figure 2.5 (e) was further label using number. Which allows the machine learning classification much easier as compare to a character of string.

After processing the data, the third stage which show in figure 2.5 (a) is to choose one of 3 machine learning classification technique to train the model for prediction. For example, decision tree classification technique chosen as the first algorithm to train the model. Then the model will be train with the will be train with different test size and then the accuracy based of the classification technique with respective test size will be tested. After the accuracy of the model has been tested, a real traffic is used as the test data to predict whether it is a malicious or normal traffic. The subsequent machine learning classification technique is then chosen to train a new model. The steps of training a model is the same as training the model using decision tree algorithm. The last stage will be evaluating the different type of machine learning classification technique. The results from different type of classification technique and different test size is plotted into a graph.

For the system design of this project, prototyping was used to design the system. The method used is prototyping, prototyping based approach will start with planning phase then to the 3 phases that will execute until the final deliverable was deliver. The three phase is analysis, design and implementation. The 3 phases that runs together is able to have a look back to the planning phase in case there are any changes to the system. A system prototype was produced when every time the 3 phases executed. After several prototype that meet the requirement then a final implementation will be

executed. Then the final phase for this approach is a system that meet the requirement. For planning phase in this project, the timeline of this project was planned and the requirement of the project was determined. The requirement of this project is to have a machine learning model that are capable to have decision making mechanism. After the planning has finished, the next phase is the 3 phases that will produce a system prototype. First to analyse the requirement, data such as the normal wireless traffic packet and malicious traffic packets was captured by using Wireshark. The captured packet will be analyse in Wireshark to identify potential threat or malicious activity. For example, when it comes to DoS attack the packet of the traffic will be huge and the amount of the packet is also a lot. The next phase design phase, a framework of the machine learning model was designed. It includes an input, the placement strategy of the system, a decision making mechanism and finally an output that produce by the decision making mechanism. After that, the next phase is implementation. In the implementation phase the machine learning model was trained by using the traffic packets that are captured from Wireshark. Python code was used to train the machine leaning model. The design framework is then turns into a code to implement the solution in real life. The 3 phases of will produce a system prototype that will become the baseline model of the system and further improvement was need to enhance the system. This is where the 3 phases will execute several times and a final version of the system prototype will be produced. After the final system prototype meets the requirement and a final implementation will be need in order to have a system which is the final phase of the system development.

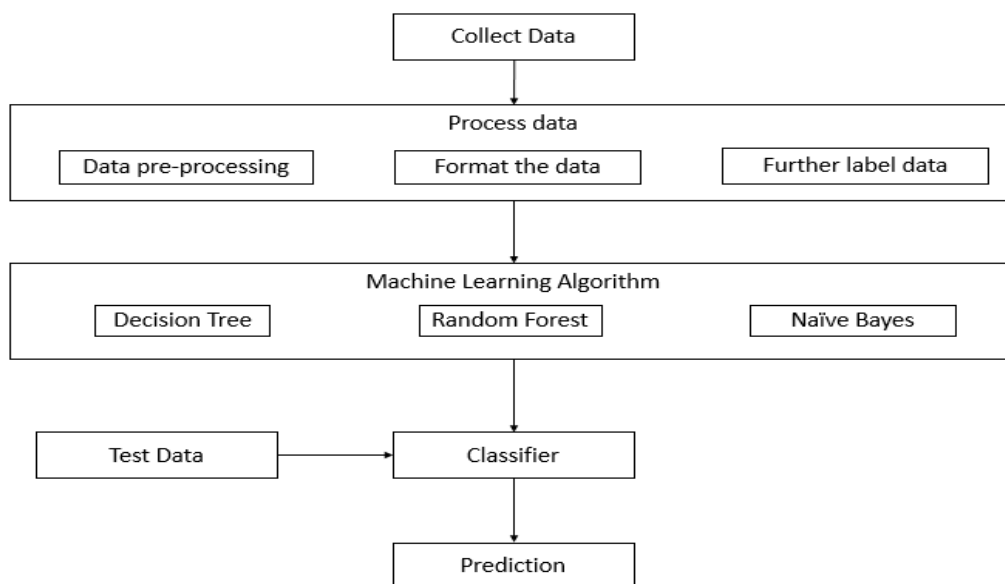


Figure 2.5 (j): Machine Learning Model

For the first machine learning classifier technique is decision tree(DT), when comes to the field of pattern recognition and data mining it is a complicated algorithm. In addition, not only database, artificial intelligent(AI), and other principles are related to it, it also brings a great impact in theoretical values. This algorithm is frequently implemented in various cases and prediction. (W.B. Zulfikar & Y.A. Gerhana & A.F. Rahmania) The second classifier which is random forest(RF), it is an ensemble leaning method which will build some randomized decision tree. After that, it will be combined and form an RF that will be used for classification and regression. (Y. Wang et. al.) The prediction of membership of a class can be done by a statistical classification method which is called naïve bayes(NB). There are a lot of works that this algorithm has been implemented such as classify eye disease, text document and image data. (W.B. Zulfikar & Y.A. Gerhana & A.F. Rahmania)

The first step to train the model for all of the classifier is to import all the related library and the data. The data used in this model is KDDTrain+.

To import related library to spyder:

*Import panda as pd*

To import related dataset:

*dataset = pd.read\_csv('KDDTrain+.csv')* (1)

The data set was imported from KDDTrain+ which is one of the data set from NSL-KDD which show in (1).

To select the feature:

```
selectedfeature = dataset.iloc[:,5:40].values (2)
```

```
condition = dataset.iloc[:,41].values (3)
```

*selectedfeature* is the independent feature (2) that used for dependent feature which is *condition* (3)

Split the data to train data and test data:

```
from sklearn.model_selection import train_test_split (4)
```

```
X_train, X_test, y_train, y_test = train_test_split(selectedfeature, condition,  
test_size=0.40, random_state = 0) (5)
```

Train\_test\_split function was imported from sklearn.model\_selection which show in (4). This function is used to split the data into training data and test data. The data was split into a few ratios for testing accuracy purpose which is (training data/test data) 0.60/0.40, 0.65/0.35, 0.70/0.30, 0.75/0.25, 0.80/0.20, 0.85/0.15 and 0.90/0.10. This ratio is set by test\_size = 0.40 which will be change to 0.35, 0.30, 0.25, 0.20,0.15 and finally 0.10. the random\_state = 0 is to ensure every time the split data is constant and 0 is the seed for the random which is in (5).

To train the model with decision tree:

```
from sklearn.tree import DecisionTreeClassifier (6)
```

```
DTclf = DecisionTreeClassifier(criterion = 'entropy', random_state = 0) (7)
```

```
DTclf.fit(X_train, y_train) (8)
```

From sklearn.tree a function called DecisionTreeClassifier was called which is decision tree algorithm in (6). *clf* is the variable name which will then call the DecisionTreeClassifier() with the parameters of criterion = 'entropy' and random\_state=0 in (7). Then the classifier *clf* was fitted with X\_train and y\_train that has been split in the train\_test\_split function. The model will be train according to the X\_train and y\_train data in (8).

To train a model with random forest algorithm:

```
from sklearn.ensemble import RandomForestClassifier (9)
```

```
RFclf=RandomForestClassifier(n_estimators=50) (10)
```

```
RFclf.fit(X_train, y_train) (11)
```

Before training the RF model equation (1) to (5) was repeated. To train a model using RF algorithm, RandomForestClassifier was imported from sklearn.ensemble (9). Then the RFclf is the variable name, the parameters of RF classifier are n\_estimator = 50 which is the number of tree equal to 50 in (10). X\_train and y\_train was fit into the clf with RF classifier to train the model.

Feature Scaling for naïve bayes:

```
from sklearn.preprocessing import StandardScaler (12)
```

```
sc = StandardScaler() (13)
```

```
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test) (14)
```

To train a model with naïve bayes network:

```
from sklearn.naive_bayes import GaussianNB (15)
```

```
GNBclf = GaussianNB() (16)
```

```
GNBclf.fit(X_train,y_train) (17)
```

Equation (1) to (5) was repeated to import the related library, data, and splitting the data. In naïve bayes, StandardScaler was imported from sklearn.preprocessing to have a feature scaling for the X\_train and X\_test. (12)(13)(14). To train the model, GaussianNB was imported from sklearn.naive\_bayes (15). GNBclf is the variable name that call the guention GaussianNB() (16). X\_train and y\_train was fitted into GNBclf to train the model.

To test the accuracy of the model:

```
DTclf.score(X_test, y_test) (18)
```

```
RFclf.score(X_test, y_test) (19)
```

$$GNBclf.score(X_{test}, y_{test}) \quad (20)$$

The accuracy of the classifier algorithm was implemented by the function score with the parameter of  $X_{test}$  and  $y_{test}$  in (18) (19) (20).

Impossible that every projects will go as what it planned at the beginning. Same goes to this project, there are also some difficulties and challenges. The difficulties that I faced during the development of this project is to train the machine learning model. The wireless traffic packet that I capture to train the machine learning model is fairly inaccurate that causes the machine learning model unable to predict the results correctly. Sometimes when I test the machine learning model, it can predict the malicious traffic but sometimes it is unable to predict the malicious traffic. Besides, to identify which parameters should be used to train the model is also one of the concern. Without the correct parameters to train the machine learning model is quite troublesome. Sometimes when I think that it is the correct parameter and I choose the parameter to train the machine learning model, my supervisor will tell me that another parameter is more suitable to train the machine learning model to have a better accuracy.

250	53.433743682	192.168.43.54	192.168.43.1	ICMP	1042 Echo (ping) request	id=0xde0e, seq=0/0, ttl=64 (reply ir
251	53.449093992	192.168.43.1	192.168.43.54	ICMP	1042 Echo (ping) reply	id=0xde0e, seq=0/0, ttl=64 (request
252	54.433959158	192.168.43.54	192.168.43.1	ICMP	1042 Echo (ping) request	id=0xde0e, seq=256/1, ttl=64 (reply
253	54.446598685	192.168.43.1	192.168.43.54	ICMP	1042 Echo (ping) reply	id=0xde0e, seq=256/1, ttl=64 (request
254	55.434647068	192.168.43.54	192.168.43.1	ICMP	1042 Echo (ping) request	id=0xde0e, seq=512/2, ttl=64 (reply
255	55.449551381	192.168.43.1	192.168.43.54	ICMP	1042 Echo (ping) reply	id=0xde0e, seq=512/2, ttl=64 (request
256	56.436216485	192.168.43.54	192.168.43.1	ICMP	1042 Echo (ping) request	id=0xde0e, seq=768/3, ttl=64 (reply
257	56.451619897	192.168.43.1	192.168.43.54	ICMP	1042 Echo (ping) reply	id=0xde0e, seq=768/3, ttl=64 (request
258	57.436831500	192.168.43.54	192.168.43.1	ICMP	1042 Echo (ping) request	id=0xde0e, seq=1024/4, ttl=64 (reply
259	57.451411979	192.168.43.1	192.168.43.54	ICMP	1042 Echo (ping) reply	id=0xde0e, seq=1024/4, ttl=64 (request
260	58.437536014	192.168.43.54	192.168.43.1	ICMP	1042 Echo (ping) request	id=0xde0e, seq=1280/5, ttl=64 (reply
261	58.450101750	192.168.43.1	192.168.43.54	ICMP	1042 Echo (ping) reply	id=0xde0e, seq=1280/5, ttl=64 (request
262	59.439055250	192.168.43.54	192.168.43.1	ICMP	1042 Echo (ping) request	id=0xde0e, seq=1536/6, ttl=64 (reply
263	59.452064730	192.168.43.1	192.168.43.54	ICMP	1042 Echo (ping) reply	id=0xde0e, seq=1536/6, ttl=64 (request
264	60.439965960	192.168.43.54	192.168.43.1	ICMP	1042 Echo (ping) request	id=0xde0e, seq=1792/7, ttl=64 (reply
265	60.466776910	192.168.43.1	192.168.43.54	ICMP	1042 Echo (ping) reply	id=0xde0e, seq=1792/7, ttl=64 (request
266	61.441205527	192.168.43.54	192.168.43.1	ICMP	1042 Echo (ping) request	id=0xde0e, seq=2048/8, ttl=64 (reply
267	61.452695611	192.168.43.1	192.168.43.54	ICMP	1042 Echo (ping) reply	id=0xde0e, seq=2048/8, ttl=64 (request
268	62.441857157	192.168.43.54	192.168.43.1	ICMP	1042 Echo (ping) request	id=0xde0e, seq=2304/9, ttl=64 (reply
269	62.453622270	192.168.43.1	192.168.43.54	ICMP	1042 Echo (ping) reply	id=0xde0e, seq=2304/9, ttl=64 (request

Figure 2.5 (k): Sample wireshark packet



192.168.43.54. the continuous ICMP request and reply is done by simple ping attack by the command `hping-3 -icmp -c 1000 -d 1000 --spooft 192.168.43.54 192.168.43.1` which is smurf attack. As we further decapsulate the ICMP packet in figure 3(l) we can clearly see the destination of the packet is VivoMobi\_dd:a6:6a and the source is NeotuneI\_1b:8c:c7. When we look deeper in the ICMP section, we are able to see that the data is 1000bytes and it is type 8 which is echo (ping) request. Figure 3(k) shows the easiest ICMP attacks in a network which are able to be captured by Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
50	34.353854085	fe80::dd45:9d07:f71...	ff02::16	ICMPv6	110	Multicast
51	34.465983565	fe80::4981:9c3b:1ff...	ff02::16	ICMPv6	90	Multicast
52	36.045678792	192.168.0.160	224.0.0.251	MDNS	91	Star
53	36.046318711	fe80::184c:9cdc:f8b...	ff02::fb	MDNS	111	Star
54	37.072658656	192.168.0.160	224.0.0.251	MDNS	91	Star
55	37.072685731	fe80::184c:9cdc:f8b...	ff02::fb	MDNS	111	Star
56	38.485675979	fe80::429b:cdff:feb...	ff02::1	ICMPv6	158	Router
57	38.497945934	fe80::dd45:9d07:f71...	ff02::16	ICMPv6	110	Multicast
58	39.292157148	fe80::dd45:9d07:f71...	ff02::16	ICMPv6	110	Multicast
59	40.082385098	192.168.0.160	224.0.0.251	MDNS	91	Star
60	40.082399178	fe80::184c:9cdc:f8b...	ff02::fb	MDNS	111	Star
61	42.887545104	fe80::429b:cdff:feb...	ff02::1	ICMPv6	158	Router
62	42.898010317	fe80::dd45:9d07:f71...	ff02::16	ICMPv6	110	Multicast
63	43.196420685	fe80::dd45:9d07:f71...	ff02::16	ICMPv6	110	Multicast
64	46.325600394	fe80::429b:cdff:feb...	ff02::1	ICMPv6	158	Router
65	46.337634589	fe80::dd45:9d07:f71...	ff02::16	ICMPv6	110	Multicast
66	46.781287054	fe80::dd45:9d07:f71...	ff02::16	ICMPv6	110	Multicast
67	49.196632407	192.168.0.160	224.0.0.251	MDNS	91	Star
68	49.196645183	fe80::184c:9cdc:f8b...	ff02::fb	MDNS	111	Star

Figure 2.5 (n): Normal traffic

Figure 3(n) shows a normal traffic in the network, this network does not contain any suspicious activities. This network is consisting of 2 clients and a unifi router. Based on figure 3(n) we are can analyse the client of the network is surfing web. Which is a normal traffic that captured by Wireshark.



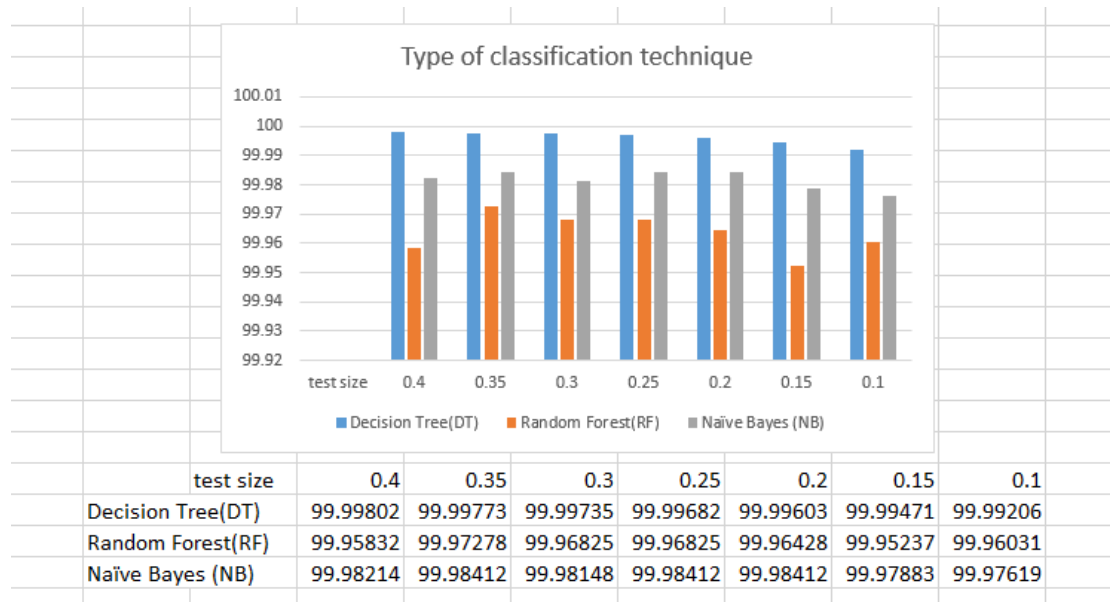


Figure 2.5(o): Result and graph of type of machine learning classification techniques and different test size

Figure 4 shows the result and graph regarding different type of machine learning classification technique with the respective test size. From figure 4 the graph shows us that RF has the least accuracy according to  $RFelf.score(X_{test}, y_{test})$ . As we can see, the test size with 0.35 has the highest accuracy. On the other hand, the test size with 0.15 has the lowest accuracy. The mean for DT with different test size was calculated which is 99.96351%. NB has the second highest of accuracy with the mean of 99.98157% and the highest accuracy is DT that have a mean of 99.9961%. In NB, the test size that have the higher accuracy is 0.20 with the score of 99.984123834094%. the test size with the least accuracy is 0.10 which has the score of 99.97619. In DT it has the highest accuracy among the other 2 algorithms. The test size that has the highest accuracy is 0.30 which has the score of 99.99735 and the test size with the least accuracy is 0.10 with the score of 99.99206. Furthermore, the mean for this 3 machine learning classification technique is calculated. The least accurate classification technique is RF which has a mean of 99.9635. The most accurate classification technique is DT which

has a mean of 99.9961 and the mean for naïve bayes is 99.9816 which makes it the second most accurate.

The method that are widely used by everyone is decision tree because it is a non-linear supervised model. Besides it is able to classify into different categories in order to make an accurate prediction from unseen data. DT model usually breaks the result into if-else statement that can be seen as a tree-like graph. Tree-like model offer a high degree of comprehension to make it flexible and easy for human and machine to learn the discovered knowledge. (Hang Yang et. al. 2014) Besides, DT is using an approach that consist of multistage decision making. DT is a consecutive model that will compare a numeric attribute against a threshold value. (Anuradha & Dr.Gaurav Gupta 2014) RF is a classifier that builds up using DT which is the basic classifier for RF. Based on the upper bound generalization error in RF, RF's classification ability can be further improve. Accuracy of single sub-based classifier should be enhanced in order to have a better accuracy of RF. (Yu Liu et. al. 2019) RF is an ensemble leaning method that is highly dependent on distinctiveness of the individual base classifier. (M. Bader-El-Den & E. Teitei & T. Perry 2017) NB consider as a statistical classification technique that able to predict probability of ownership of a class. (W.B. Zulfikar, Y.A. Gerhana, A.F. Rahmania 2018) Besides, it is also a probabilistic analyser which are capable to do basic stuff. It is able to sum all the frequencies and values of data to compute a set of probabilities. The model train using NB able to allow every attribute to contribute in the final decision.(F. Harahap et. al. 2018)

## Chapter 3: System Design

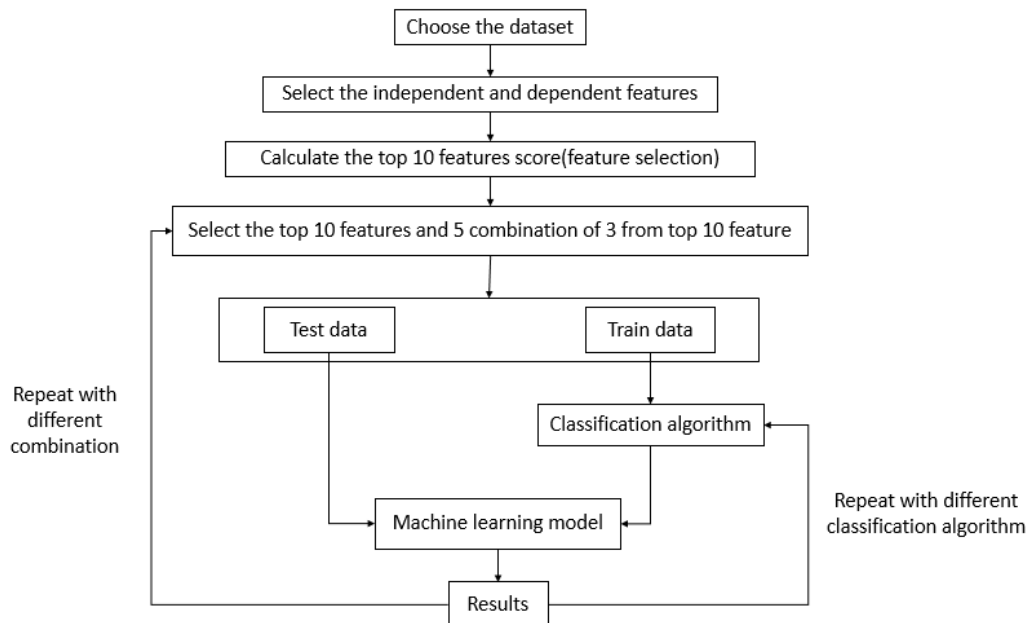


Figure 3(a): Work flow of this research

There is few enhancement was made in the following code as compare to the previous work. The code from preliminary work was enhanced. The following are the enhancement work.

The first step to train the model for all of the classifier is to import all the related library and the data. The wired dataset used to train the following model is KDDTrain+ and the dataset that used to test the model is KDDTest+. There is a total of 3 files which is IDS.py, train.py and report.py. This 3 files is used to train the model, predict using test data and calculate and print out the results for each of the classifier.

For the first file to be discuss of this project which is the train.py. It is a file which used to train each of the machine learning model. From (1) to (6) it is the library from spyder. (7) is a self-define library which will be discuss later on.

First, in the file we import all the necessary library.

```
from sklearn import svm (1)
```

```
from sklearn.naive_bayes import GaussianNB (2)
```

```
from sklearn.model_selection import train_test_split (3)
```

```
from sklearn.ensemble import RandomForestClassifier (4)
```

```
from sklearn.tree import DecisionTreeClassifier (5)
```

```
from sklearn.metrics import confusion_matrix (6)
```

```
from report import report (7)
```

After all of the related library has been imported, the next step is to define the function for each of the model. The first model to be define is decision tree, the function name will be ModelDT(combination, bestTestData, condition, expected) in (8). The function needs four parameters which is combination, bestTestData, condition and expected. The parameter combination is the random combination of three out of the top 10 features from the train data. For bestTestData, it is the same as combination but it has the feature from test data. Then the data will split into train data/test data (0.60/0.40) in (9). The decision tree classifier was assigned to a variable name *modelDT* and the parameter of *DecisionTreeClassifier()* is kept default in (10). The model is then fit with *X\_train* and *y\_train* (11) which is the result from splitting the data (9). The trained model is used to predict test data which shows in (12). A confusion matrix is plotted in (13). The results from confusion matrix which is true positive (*DTtp*), true negative (*DTtn*), false positive (*DTfp*) and false negative (*DTfn*) was assigned to the respective variable in (13). The results from (13) is fitted into another self-define function to calculate accuracy, precision, recall and f1 score and the value for each true negative, true positive, false negative and false positive was print out.

```
def ModelDT(combination,bestTestData,condition,expected): (8)
```

```
    X_train, X_test, y_train, y_test = train_test_split(combination,
```

```
    condition, test_size=0.40, random_state=0) (9)
```

```
    modelDT = DecisionTreeClassifier() (10)
```

```
    modelDT.fit(X_train,y_train) (11)
```

```
    predictedDT=modelDT.predict(bestTestData) (12)
```

```
    DTtn, DTfp, DTfn, DTtp=confusion_matrix(expected,  
    predictedDT).ravel() (13)
```

$$report(DTtp,DTfp,DTfn,DTtn,predictedDT,expected) \quad (14)$$

The next function in the file is `ModelRF(combination, bestTestData, condition, expected)` in (15). The definition for each of the parameter in `ModelRF` is same as `ModelDT`. The dataset is split into train data and test data (0.60/0.40) in (16). `modelRF` is the variable name that used to assign `RandomForestClassifier(n_estimators=100)` which is the random forest classifier. The split data `X_train` and `y_train` is fit into the random forest model. The trained model is used to predict test data (20). A confusion matrix is plotted to see the results of the prediction of the model. The results were assigned to the respective variable which is true positive (`RFtp`), true negative (`RFtn`), false positive (`RFfp`) and false negative (`RFfn`) (21). The accuracy, precision, recall and f1 score is calculated in (22). Besides, the results of true positive, true negative, false positive and false negative were print out along with accuracy, precision, recall and f1 score in (22).

$$def ModelRF(combination, bestTestData, condition, expected): \quad (15)$$

$$X\_train, X\_test, y\_train, y\_test = train\_test\_split(combination, \quad (16)$$

$$condition, test\_size=0.40, random\_state=0)$$

$$modelRF = RandomForestClassifier(n\_estimators=100) \quad (17)$$

$$modelRF.fit(X\_train, y\_train) \quad (18)$$

$$predictedRF = modelRF.predict(bestTestData) \quad (20)$$

$$RFtn, RFfp, RFfn, RFtp = confusion\_matrix(expected, \quad (21)$$

$$predictedRF).ravel()$$

$$report(RFtp, RFfp, RFfn, RFtn, predictedRF, expected) \quad (22)$$

The third function in the file is `ModelNB(combination, bestTestData, condition, expected)` which is used to train naïve bayes model in (23). The definition for each of the parameter in `ModelNB` is same as previously self-define function. The dataset is split into train data and test data (0.60/0.40) in (24). `modelNB` is the variable name that used to assign `GaussianNB()` which is the naïve bayes classifier it used the default parameter in (25). The split data `X_train` and `y_train` is fit

into the naïve bayes model (26). The trained model is used to predict test data (27). A confusion matrix is plotted to see the results of the prediction of the model. The results were assigned to the respective variable which is true positive (*NBtp*), true negative (*NBtn*), false positive (*NBtp*) and false negative (*NBfn*) (28). All of the results such as true positive, true negative, false positive, false negative, accuracy, precision, recall and f1 score is calculated and print out by using this function in (29).

```
def ModelNB(combination,bestTestData,condition,expected): (23)
```

```
    X_train, X_test, y_train, y_test = train_test_split(combination,
condition, test_size=0.40, random_state=0) (24)
```

```
    modelNB = GaussianNB() (25)
```

```
    modelNB.fit(X_train,y_train) (26)
```

```
    predictedNB=modelNB.predict(bestTestData) (27)
```

```
    NBtn, Nbfp, NBfn, NBtp = confusion_matrix(expected,
predictedNB).ravel() (28)
```

```
    report(NBtn,Nbfp,NBfn,NBtp,predictedNB,expected) (29)
```

The final function in the file which is `ModelSVC(combination, bestTestData,condition,expected)` which is used to train support vector machine model in (30). The definition for each of the parameter in `ModelSVC` is same as previously self-define function. The dataset is split into train data and test data (0.60/0.40) in (31). `modelSVC` is the variable name that used to assign `svm.SVC()` which is the support vector machine classifier it used the default parameter in (32). The split data `X_train` and `y_train` is fit into the support vector machine model (33). The trained model is used to predict test data (34). A confusion matrix is plotted to see the results of the prediction of the model. The results were assigned to the respective variable which is true positive (*NBtp*), true negative (*NBtn*), false positive (*NBtp*) and false negative (*NBfn*) (35). All of the results such as true positive, true negative, false positive, false negative, accuracy, precision, recall and f1 score is calculated and print out by using this function (36).

```
def ModelSVC(combination,bestTestData,condition,expected): (30)
```

```
    X_train, X_test, y_train, y_test = train_test_split(combination,
```

```
condition, test_size=0.40, random_state=0) (31)
```

```
modelSVC = svm.SVC() (32)
```

```
modelSVC.fit(X_train,y_train) (33)
```

```
predictedSVC=modelSVC.predict(bestTestData) (34)
```

```
SVCtn, SVCfp, SVCfn, SVCtp = confusion_matrix(expected,  
predictedSVC).ravel() (35)
```

```
report(SVCtn,SVCfp,SVCfn,SVCtp,predictedNB,expected) (36)
```

The second file is report.py which is a self-defined library to calculate and display the results.

First, there are two libraries were imported.

```
from sklearn.metrics import (precision_score,  
recall_score,f1_score,accuracy_score,mean_squared_error,  
mean_absolute_error) (1)
```

```
from sklearn.metrics import classification_report (2)
```

The function that calculate and display all of the results is *report(truePositive, falsePositive,falseNegative,pred,train)* (3). It takes 6 parameters into it, which is *truePositive, falsePositive, falseNegative ,pred* and *train*. *pred* is the prediction of the model which made by a trained model, and *train* is the dependent feature of the test data. Then the parameter is assigned to respective variable (4). The accuracy score is calculated by using *accuracy\_score(train,pred)* in (5). To calculate the accuracy score, the formula used is  $(TP+TN)/(TP+FP+FN+TN)$  where TP is true positive, TN is true negative, FP is false positive and FN is false negative. Accuracy is the ratio of correctly predicted to the total number. The following is to calculate recall which use *recall\_score(train,pred, average="binary")*. The formula used to calculate recall score is  $TP/(TP+FN)$ . Recall is the ratio of correctly predicted to the total of correctly predicted packet and normal packet. Besides, the precision score of the model is also calculated with the formula  $TP/(TP+FP)$  in (7). Finally, the f1 score which is the average for precision and recall. To calculate f1 score, *f1\_score(train,pred,*

*average="binary"*) is used (8). The formula to calculate f1 score is  $2(\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$ . Besides, the true positive rate and false positive rate was also calculated in (9) and (10). True positive, true negative, false positive and false negative was printed out in (11). The results from the calculation of accuracy, precision, recall and f1 score was print out in (9), (10), (11) and (12). A summary of each classes is printed out in (14) which will be more specific about the results on normal and attack classes. Finally, accuracy, recall, precision,f1, true positive rate and false positive rate was return by this function.

```
trueNegative,pred,train)def report(truePositive,falsePositive,falseNegative,
```

```
trueNegative,pred,train): (3)
```

```
tp,fp,fn,tn = truePositive,falsePositive,falseNegative,trueNegative (4)
```

```
accuracy = accuracy_score(train, pred) (5)
```

```
recall = recall_score(train,pred, average = "binary") (6)
```

```
precision = precision_score(train, pred, average="binary") (7)
```

```
f1 = f1_score(train, pred, average="binary") (8)
```

```
tpr=tp/(tp+fn) (9)
```

```
fpr=fp/(fp+tn) (10)
```

```
print("TP:",tp,"\nFP:",fp,"\nFN:",fn,"\nTN:",tn,"\n") (11)
```

```
print("Accuracy : %.3f "%accuracy) (12)
```

```
print("Precision : %.3f" %precision) (13)
```

```
print("Recall: %.3f"%recall) (14)
```

```
print("F1score: %.3f"%f1) (15)
```

```
print("\n") (16)
```

```
print(classification_report(train, pred)) (17)
```

```
return (accuracy,recall,precision,f1,tpr,fpr) (18)
```



The final file which is the IDS.py, it is the main function to call all of the related library and function to train the model, predict with the model and display the results.

First, the necessary libraries were imported.

```
import pandas as pd (1)
```

```
from sklearn import metrics (2)
```

```
from sklearn.feature_selection import SelectKBest (3)
```

```
from sklearn.feature_selection import chi2 (4)
```

```
from sklearn.feature_selection import f_classif (5)
```

```
from train import * (6)
```

The second step will be read the dataset from their own respective files. The train data and test data for NSL-KDD is *traindata* and *testdata* in (7) and (8). While, *wirelessTrainData* and *wirelessTestData* is the train and test data for NaBIoT which is wireless dataset.

```
traindata = pd.read_csv('KDDTrain+.csv') (7)
```

```
testdata = pd.read_csv('KDDTest+.csv') (8)
```

```
wirelessTrainData = pd.read_csv('wireless.csv') (9)
```

```
wirelessTestData = pd.read_csv('wirelessTest1.csv') (10)
```

The label of each attack packet was changed to 1 attack and normal packet change to 0 in (11). Besides, the service, protocol and flag was change to numeric value. The following (12), (13), (14) and (15) are the codes that change the data to respective numeric value in train data.

```
attack = {
```

```
    'back' : 1, 'buffer_overflow' : 1, 'ftp_write' : 1, 'guess_passwd' : 1,
```

```
    'imap' : 1, 'ipsweep' : 1, 'land' : 1, 'loadmodule' : 1, 'nmap' : 1,
```

```
    'multihop' : 1, 'neptune' : 1, 'perl' : 1, 'phf' : 1, 'pod' : 1,
```

```
    'portsweep' : 1, 'rootkit' : 1, 'satan' : 1, 'smurf':1, 'teardrop' : 1,
```

```

    'spy' : 1, 'warezclient' : 1, 'warezmaster' : 1, 'normal':0
}

traindata = traindata.replace({'Attack':attack})

```

(11)

```

Service = {
    'other' : 1, 'link' : 2, 'netbios_ssn' : 3, 'smtp' : 4, 'netstat' : 5, 'ctf' : 6,
    'ntp_u' : 7, 'harvest' : 8, 'efs' : 9, 'klogin' : 10, 'systat' : 11, 'exec' : 12,
    'nntp' : 13, 'pop_3' : 14, 'printer' : 15, 'vmnet' : 16, 'netbios_ns' : 17,
    'urh_i' : 18, 'ssh' : 19, 'http_8001' : 20, 'iso_tsap' : 21, 'aol' : 22,
    'sql_net' : 23, 'shell' : 24, 'supdup' : 25, 'auth' : 26, 'whois' : 27,
    'discard' : 28, 'sunrpc' : 29, 'urp_i' : 30, 'rje' : 31, 'ftp' : 32,
    'daytime' : 33, 'domain_u' : 34, 'pm_dump' : 35, 'time' : 36,
    'hostnames' : 37, 'name' : 38, 'ecr_i' : 39, 'ecr_i' : 39, 'bgp' : 40,
    'telnet' : 41, 'domain' : 42, 'ftp_data' : 43, 'nntp' : 44, 'courier' : 45,
    'finger' : 46, 'uucp_path' : 47, 'X11' : 48, 'imap4' : 49, 'mtp' : 50,
    'login' : 51, 'tftp_u' : 52, 'kshell' : 53, 'private' : 54, 'http_2784' : 55,
    'echo' : 56, 'http' : 57, 'ldap' : 58, 'tim_i' : 59, 'netbios_dgm' : 60, 'uucp' : 61,
    'eco_i' : 62, 'remote_job' : 63, 'IRC' : 64, 'http_443' : 65,
    'red_i' : 66, 'Z39_50' : 67, 'pop_2' : 68, 'gopher' : 69, 'csnet_ns' : 70
}

```

```

traindata = traindata.replace({'Service':Service})

```

(12)

```

Protocol = {
    'icmp' : 1, 'tcp' : 2, 'udp' : 3
}

traindata = traindata.replace({'Protocol':Protocol})

```

(13)

```

Flag = {
    'OTH' : 1, 'S1':2,'S2':3,'RSTO':4,'RSTR':5,
    'RSTOSO':6,'SF':7,'SH':8,'REJ':9,'S0':10,'S3':11
}

traindata = traindata.replace({'Flag':Flag})

```

(14)

The following codes are used to change the data to respective numeric values for test data.

```

attack = {
    'back' : 1,'buffer_overflow' : 1,
    'ftp_write' : 1, 'guess_passwd' : 1,
    'imap' : 1, 'ipsweep' : 1,'land' : 1, 'loadmodule' : 1,
    'multihop' : 1, 'neptune' : 1,'nmap' : 1,'perl' : 1, 'phf' :1,'pod' : 1,
    'portsweep' : 1,'rootkit' : 1,'satan' : 1,'smurf':1,'spy' : 1,
    'teardrop' : 1,'warezclient' : 1,'warezmaster' :1 ,
    'normal':0,'saint':1,'mscan':1, 'apache2':1,
    'snmpgetattack':1,'processtable':1,'httptunnel':1,'snmpguess':1,
    'mailbomb':1,'named':1,'ps':1,'sendmail':1,'xterm':1, 'worm':1, 'xlock':1,
    'xsnoop':1, 'sqlattack':1, 'udpstorm':1
}

testdata = testdata.replace({'Attack':attack})

```

(15)

```

Service = {
    'other' : 1, 'link' : 2, 'netbios_ssn' :3, 'smtp' : 4, 'netstat' : 5, 'ctf' : 6,
    'ntp_u' : 7, 'harvest' : 8, 'efs' : 9, 'klogin' : 10, 'systat' : 11, 'exec' : 12,
    'nntp' : 13, 'pop_3' : 14, 'printer' : 15, 'vmnet' : 16, 'netbios_ns' : 17,

```

```

'urh_i' : 18, 'ssh' : 19, 'http_8001': 20, 'iso_tsap' :21, 'aol' : 22,
'sql_net' : 23, 'shell' : 24, 'supdup' : 25, 'auth' : 26, 'whois' : 27,
'discard' : 28, 'sunrpc' : 29, 'urp_i' : 30, 'rje' : 31, 'ftp' : 32,
'daytime' : 33, 'domain_u':34, 'pm_dump' : 35, 'time' : 36,
'hostnames':37, 'name' : 38, 'ecr_i' :39, 'ecr_i' :39, 'bgp' : 40,
'telnet':41, 'domain':42, 'ftp_data':43, 'nnsf':44, 'courier':45,
'finger':46, 'uucp_path':47, 'X11':48, 'imap4':49, 'mtp':50,
'login':51, 'tftp_u':52, 'kshell':53, 'private':54, 'http_2784':55,
'echo':56, 'http':57, 'ldap':58, 'tim_i':59, 'netbios_dgm':60, 'uucp':61,
'eco_i':62, 'remote_job':63, 'IRC':64, 'http_443':65,
'red_i':66, 'Z39_50':67, 'pop_2':68, 'gopher':69, 'csnet_ns':70
}

```

```
testdata = testdata.replace({'Service':Service}) (16)
```

```

Protocol = {
    'icmp':1, 'tcp':2, 'udp':3
}

```

```
testdata = testdata.replace({'Protocol':Protocol}) (17)
```

```

Flag = {
    'OTH' : 1, 'S1':2, 'S2':3, 'RSTO':4, 'RSTR':5,
    'RSTOSO':6, 'SF':7, 'SH':8, 'REJ':9, 'S0':10, 'S3':11
}

```

```
testdata = testdata.replace({'Flag':Flag}) (18)
```

After the train dataset was imported, the independent feature and dependent feature was selected using the following code (7) and (8). The independent feature and dependent feature of test dataset by using the following code (9) and (10). The

independent feature which is *selectedfeature* (7) and the dependent feature which is *condition* (9). (9) is the independent for test data and (10) is the dependent feature of test data.

To select the feature:

*selectedfeature = traindata.iloc[:, :-2]* (19)

*condition = traindata.iloc[:, -2]* (20)

*test = testdata.iloc[:, :]* (21)

*condition = testdata.iloc[:, -2]* (22)

To find out the top 10 scores of the feature for NSL-KDD. SelectKBest function from `sklearn.feature_selection` by using ANOVA to calculate the top 10 scores in (24). In (29), the data frame column is named as 'Feature' and 'Score'. Then the top 10 largest scores were printed in a table in (30).

*dfcolumns = pd.DataFrame(selectedfeature.columns)* (23)

*bestfeatures = SelectKBest(score\_func=f\_classif, k=10)* (24)

*fit = bestfeatures.fit(selectedfeature, condition)* (25)

*dfscores = pd.DataFrame(fit.scores\_)* (26)

*dfcolumns = pd.DataFrame(traindata.columns)* (27)

*featureScores = pd.concat([dfcolumns, dfscores], axis=1)* (28)

*featureScores.columns = ['Feature', 'Score']* (29)

*print(featureScores)* (30)

*print(featureScores.nlargest(10, 'Score'))* (31)

Besides, the top 10 scores of the feature for NaBIoT. SelectKBest function from `sklearn.feature_selection` by using ANOVA to calculate the top 10 scores in (33). In (38), the data frame column is named as 'Feature' and 'Score'. Then the top 10 largest scores were printed in a table in (40).

*dfcolumns = pd.DataFrame(selectedFeatureWireless.columns)* (32)

```
bestfeatures = SelectKBest(score_func=f_classif, k=10) (33)
```

```
fit = bestfeatures.fit(selectedFeatureWireless,condition) (34)
```

```
dfscores = pd.DataFrame(fit.scores_) (35)
```

```
dfcolumns = pd.DataFrame(wirelessTrainData.columns) (36)
```

```
featureScores = pd.concat([dfcolumns,dfscores],axis=1) (37)
```

```
featureScores.columns = ['Feature','Score'] (38)
```

```
print(featureScores) (39)
```

```
print(featureScores.nlargest(10,'Score')) (40)
```

The top 10 features score was determined. Therefore, the top 10 features were selected to train the model and a combination of 3 from the top 10 features.

```
#top n=10 features from K features
```

```
bestSelectedFeature = traindata.iloc[:,[28,32,33,11,38,37,24,25,4,22]] (41)
```

```
bestTestData = testdata.iloc[:,[28,32,33,11,38,37,24,25,4,22]] (42)
```

```
#no.1 combination of 3 from K features
```

```
bestSelectedFeatureA = traindata.iloc[:,[28,33,32]] (43)
```

```
bestTestDataA = testdata.iloc[:,[28,33,32]] (44)
```

```
#no.2 combination of 3 from K features
```

```
bestSelectedFeatureB = traindata.iloc[:,[38,11,22]] (45)
```

```
bestTestDataB = testdata.iloc[:,[38,11,22]] (46)
```

```
#no.3 combination of 3 from K features
```

```
bestSelectedFeatureC = traindata.iloc[:,[25,3,24]] (47)
```

```
bestTestDataC = testdata.iloc[:,[25,3,24]] (48)
```

*#no.4 combination of 3 from K features*

$$\text{bestSelectedFeatureD} = \text{traindata.iloc[:, [25, 32, 3]]} \quad (49)$$

$$\text{bestTestDataD} = \text{testdata.iloc[:, [25, 32, 3]]} \quad (50)$$

*#no.5 combination of 3 from K features*

$$\text{bestSelectedFeatureE} = \text{traindata.iloc[:, [37, 32, 3]]} \quad (51)$$

$$\text{bestTestDataE} = \text{testdata.iloc[:, [37, 32, 3]]} \quad (52)$$

After the top 10 features for NaBIoT was determined. The top 10 features of the dataset were selected and also the combination of 3 from top 10 features.

*#top n=10 features from K features*

$$\begin{aligned} \text{bestSelectedFeature} &= \text{wirelessTrainData.iloc} \\ &[:, [28, 32, 33, 11, 38, 37, 24, 25, 4, 22]] \end{aligned} \quad (53)$$

$$\begin{aligned} \text{bestTestData} &= \text{wirelessTestData.iloc} \\ &[:, [28, 32, 33, 11, 38, 37, 24, 25, 4, 22]] \end{aligned} \quad (54)$$

*#no.1 combination from 3 from K features*

$$\text{bestSelectedFeatureA} = \text{wirelessTrainData.iloc[:, [9, 24, 6]]} \quad (55)$$

$$\text{bestTestDataA} = \text{wirelessTestData.iloc[:, [9, 24, 6]]} \quad (56)$$

*#no.2 combination from 3 from K features*

$$\text{bestSelectedFeatureB} = \text{wirelessTrainData.iloc[:, [21, 18, 6]]} \quad (57)$$

$$\text{bestTestDataB} = \text{wirelessTestData.iloc[:, [21, 18, 6]]} \quad (58)$$

*#no.3 combination from 3 from K features*

$$\text{bestSelectedFeatureC} = \text{wirelessTrainData.iloc[:, [3, 18, 0]]} \quad (59)$$

$$\text{bestTestDataC} = \text{wirelessTestData.iloc[:, [3, 18, 0]]} \quad (60)$$

*#no.4 combination from 3 from K features*

*bestSelectedFeatureD=wirelessTrainData.iloc[:,[9,27,15]]* (61)

*bestTestDataD = wirelessTestData.iloc[:,[9,27,15]]* (62)

*#no.5 combination from 3 from K features*

*bestSelectedFeatureE=wirelessTrainData.iloc[:,[24,21,0]]* (63)

*bestTestDataE = wirelessTestData.iloc[:,[24,21,0]]* (64)

With the selected features, the selected features were fit into training function which discuss earlier. The first model to be trained is decision tree model. The function used is ModelDT(combination, bestTestData, condition, expected). First, the top 10 features of the dataset was fitted, ModelDT(bestSelectedFeature, bestTestData, condition, expected). Then, the bestTestData is changed to bestTestDataA, bestTestDataB, bestTestDataC, bestTestDataD, bestTestDataE and bestSelectedFeature is changed to bestSelectedFeatureA, bestSelectedFeatureB, bestSelectedFeatureC, bestSelectedFeatureD and bestSelectedFeatureE. After the model is successfully trained, the results are return by each of the function and store into the respective variable. After that, the respective results were stored in a table in (79). In (80) the data from (79) is fitted into the data frame to have better visualization. From (81) to (84) these are the code that used to plot the graph according to the results for each of the combination for one classifier. The same steps were repeated in other naïve bayes, random forest and support vector machine and different dataset which is NaBIoT dataset in this case.

*print("-----NSL-KDD dataset-----")* (65)

*#NSL-KDD*

*print("-----DT-----")*(66)

*print("-----Top 10 Features-----")* (67)

*DTtpr, DTfpr, acc, recall,precision,f1 = ModelDT*

*(bestSelectedFeature,bestTestData,condition, expected)* (68)



*print("-----Combination A-----")* (69)

*DTtprA,DTfprA, accA,recallA,precisionA,f1\_A=*

*ModelDT(bestSelectedFeatureA,bestTestDataA,condition, expected)* (70)

*print("-----Combination B-----")* (71)

*DTtprB,DTfprB, accB,recallB,precisionB,f1\_B =*

*ModelDT(bestSelectedFeatureB,bestTestDataB,condition, expected)* (72)

*print("-----Combination C-----")* (73)

*DTtprC,DTfprC,accC,recallC,precisionC,f1\_C=*

*ModelDT(bestSelectedFeatureC,bestTestDataC,condition, expected)* (74)

*print("-----Combination D-----")* (75)

*DTtprD,DTfprD,accD,recallD,precisionD,f1\_D=*

*ModelDT(bestSelectedFeatureD,bestTestDataD,condition, expected)* (76)

*print("-----Combination E-----")* (77)

*DTtprE,DTfprE,accE,recallE,precisionE,f1\_E=*

*ModelDT(bestSelectedFeatureE,bestTestDataE,condition, expected)* (78)

*data = {'Top 10 Features','Combination A','Combination B',*

*'Combination C', 'Combination D', 'Combination E'],*

*'False Positive Rate':[DTfpr,DTfprA,DTfprB,DTfprC,DTfprD,DTfprE],*

*'True Positive Rate':[DTtpr,DTtprA,DTtprB,DTtprC,DTtprD,DTtprE],*

*'Accuracy':[acc,accA,accB,accC,accD,accE],*

*'Recall':[recall,recallA,recallB,recallC,recallD,recallE],*

*'Precision':[precision,precisionA,precisionB,*

*precisionC,precisionD,precisionE],*

```
'F1 Score':[f1,f1_A,f1_B,f1_C,f1_D,f1_E] } (79)
```

```
df = pd.DataFrame (data, columns = ['False Positive Rate',
'True Positive Rate','Accuracy','Recall','Precision','F1 Score']) (80)
```

```
plt.plot([DTfpr,DTfprA,DTfprB,DTfprC,DTfprD,DTfprE],'go-',
[DTtpr,DTtprA,DTtprB,DTtprC,DTtprD,DTtprE],'ro-',
[acc,accA,accB,accC,accD,accE],'bo-',
[recall,recallA,recallB,recallC,recallD,recallE],'co-',
[precision,precisionA,precisionB,precisionC,precisionD,precisionE],'mo-',
[f1,f1_A,f1_B,f1_C,f1_D,f1_E],'yo-') (81)
```

```
plt.title("Results of Decision Tree Model(NSL-KDD)") (82)
```

```
plt.xlabel("False Positive Rate = Green \n True Positive Rate = Red\n
Accuracy Score= Blue\nRecall = Cyan\nPrecision = Magenta\n
F1 Score = Yellow") (83)
```

```
plt.show() (84)
```

```
print("-----NB-----")
print(" ")
print("-----Top 10 Features-----") (85)
```

```
NBtpr, NBfpr, acc, recall,precision,f1=
ModelNB(bestSelectedFeature,bestTestData,condition, expected) (86)
```

```
print("-----Combination A-----") (87)
```

```
NBtprA, NBfprA, accA, recallA,precisionA,f1_A=
ModelNB(bestSelectedFeatureA,bestTestDataA,condition, expected) (88)
```

```
print("-----Combination B-----") (89)
```

```
NBtprB, NBfprB, accB, recallB,precisionB,f1_B=
```

*ModelNB(bestSelectedFeatureB,bestTestDataB,condition, expected)* (90)

*print("-----Combination C-----")* (91)

*NBtprC, NBfprC, accC, recallC,precisionC,f1\_C=*

*ModelNB(bestSelectedFeatureC,bestTestDataC,condition, expected)* (92)

*print("-----Combination D-----")* (93)

*NBtprD, NBfprD, accD, recallD,precisionD,f1\_D=*

*ModelNB(bestSelectedFeatureD,bestTestDataD,condition, expected)* (94)

*print("-----Combination E-----")* (95)

*NBtprE, NBfprE, accE, recallE,precisionE,f1\_E=*

*ModelNB(bestSelectedFeatureE,bestTestDataE,condition, expected)* (96)

*data = {'Top 10 Features','Combination A','Combination B',*

*'Combination C', 'Combination D', 'Combination E'],*

*'False Positive Rate':[NBtpr,NBfprA,NBfprB,NBfprC,NBfprD,NBfprE],*

*'True Positive Rate':[NBtpr,NBtprA,NBtprB,NBtprC,NBtprD,NBtprE],*

*'Accuracy':[acc,accA,accB,accC,accD,accE],*

*'Recall':[recall,recallA,recallB,recallC,recallD,recallE],*

*'Precision':[precision,precisionA,precisionB,precisionC,*

*precisionD,precisionE],*

*'F1 Score':[f1,f1\_A,f1\_B,f1\_C,f1\_D,f1\_E] }* (97)

*df = pd.DataFrame (data, columns = ['False Positive Rate',*

*'True Positive Rate','Accuracy','Recall','Precision','F1 Score'])* (98)

*plt.plot([NBfpr,NBfprA,NBfprB,NBfprC,NBfprD,NBfprE], 'go-',*

*[NBtpr,NBtprA,NBtprB,NBtprC,NBtprD,NBtprE], 'ro-',*

*[acc,accA,accB,accC,accD,accE], 'bo-',*

*[recall,recallA,recallB,recallC,recallD,recallE], 'co-',*

```
[precision,precisionA,precisionB,precisionC,precisionD,precisionE], 'mo-',
[f1,f1_A,f1_B,f1_C,f1_D,f1_E], 'yo-') (99)
```

```
plt.title("Results of Naive Bayes Model(NSL-KDD)") (100)
```

```
plt.xlabel("False Positive Rate = Green \n True Positive Rate = Red\n
Accuracy Score= Blue\nRecall = Cyan\nPrecision = Magenta\n
F1 Score = Yellow") (101)
```

```
plt.show() (102)
```

```
print("-----RF-----") (103)
```

```
print("-----Top 10 Features-----") (104)
```

```
RFtpr, RFfpr, acc, recall,precision,f1=
ModelRF(bestSelectedFeature,bestTestData,condition, expected) (105)
```

```
print("-----Combination A-----") (106)
```

```
RFtprA, RFfprA, accA, recallA,precisionA,f1_A=
ModelRF(bestSelectedFeatureA,bestTestDataA,condition, expected) (107)
```

```
print("-----Combination B-----") (108)
```

```
RFtprB, RFfprB, accB, recallB,precisionB,f1_B=
ModelRF(bestSelectedFeatureB,bestTestDataB,condition, expected) (109)
```

```
print("-----Combination C-----") (110)
```

```
RFtprC, RFfprC, accC, recallC,precisionC,f1_C=
ModelRF(bestSelectedFeatureC,bestTestDataC,condition, expected) (111)
```

```
print("-----Combination D-----") (112)
```

```
RFtprD, RFfprD, accD, recallD,precisionD,f1_D=
ModelRF(bestSelectedFeatureD,bestTestDataD,condition, expected) (113)
```

```
print("-----Combination E-----") (114)
```

```
RFtprE, RFfprE, accE, recallE,precisionE,f1_E=
ModelRF(bestSelectedFeatureE,bestTestDataE,condition, expected) (115)
```

```
data = {'':['Top 10 Features','Combination A','Combination B',
'Combination C', 'Combination D', 'Combination E'],
'False Positive Rate':[RFtpr,RFfprA,RFfprB,RFfprC,RFfprD,RFfprE],
'True Positive Rate':[RFtpr,RFtprA,RFtprB,RFtprC,RFtprD,RFtprE],
'Accuracy':[acc,accA,accB,accC,accD,accE],
'Recall':[recall,recallA,recallB,recallC,recallD,recallE],
'Precision':[precision,precisionA,precision
,precisionC,precisionD,precisionE],
'F1 Score':[f1,f1_A,f1_B,f1_C,f1_D,f1_E] } (116)
```

```
df = pd.DataFrame (data, columns = ['','False Positive Rate',
'True Positive Rate','Accuracy','Recall','Precision','F1 Score']) (117)
```

```
plt.plot([RFtpr,RFfprA,RFfprB,RFfprC,RFfprD,RFfprE], 'go-',
[RFtpr,RFfprA,RFfprB,RFfprC,RFfprD,RFfprE], 'ro-',
[acc,accA,accB,accC,accD,accE], 'bo-',
[recall,recallA,recallB,recallC,recallD,recallE], 'co-',
[precision,precisionA,precisionB,precisionC,precisionD,precisionE], 'mo-',
[f1,f1_A,f1_B,f1_C,f1_D,f1_E], 'yo-') (118)
```

```
plt.title("Results of Random Forest Model(NSL-KDD)") (119)
```

```
plt.xlabel("False Positive Rate = Green \n True Positive Rate = Red\n
Accuracy Score= Blue\nRecall = Cyan\nPrecision = Magenta\n
F1 Score = Yellow") (120)
```

```
plt.show() (121)
```

*print("-----SVC-----")* (122)

*print(" ")*

*print("-----Top 10 Features-----")* (123)

*SVCtpr, SVCfpr, acc, recall,precision,f1=*

*ModelSVC(bestSelectedFeature,bestTestData,condition, expected)* (124)

*print("-----Combination A-----")* (125)

*SVCtprA, SVCcfprA, accA, recallA,precisionA,f1\_A=*

*ModelSVC(bestSelectedFeatureA,bestTestDataA,condition, expected)* (126)

*print("-----Combination B-----")* (127)

*SVCctpr, SVCfprB, accB, recallB,precisionB,f1\_B=*

*ModelSVC(bestSelectedFeatureB,bestTestDataB,condition, expected)* (128)

*print("-----Combination C-----")* (129)

*SVCtprC, SVCfprC, accC, recallC,precisionC,f1\_C=*

*ModelSVC(bestSelectedFeatureC,bestTestDataC,condition, expected)* (130)

*print("-----Combination D-----")* (131)

*SVCtprD, SVCfprD, accD, recallD,precisionD,f1\_D=*

*ModelSVC(bestSelectedFeatureD,bestTestDataD,condition, expected)* (132)

*print("-----Combination E-----")* (133)

*SVCtprE, SVCfprE, accE, recallE,precisionE,f1\_E=*

*ModelSVC(bestSelectedFeatureE,bestTestDataE,condition, expected)* (134)

*data = {'Top 10 Features','Combination A','Combination B',*

*'Combination C', 'Combination D', 'Combination E'],*

*'False Positive Rate':[SVCtpr,SVCfprA,SVCfprB,SVCfprC,*

*SVCfprD,SVCfprE],*

```

'True Positive Rate':[SVCtpr,SVCtprA,SVCtprB,SVCtprC,
SVCtprD,SVCtprE],
'Accuracy':[acc,accA,accB,accC,accD,accE],
'Recall':[recall,recallA,recallB,recallC,recallD,recallE],
'Precision':[precision,precisionA,precisionB,precisionC,
precisionD,precisionE],
'F1 Score':[f1,f1_A,f1_B,f1_C,f1_D,f1_E]}

```

(135)

```

df = pd.DataFrame (data, columns = ['','False Positive Rate',
'True Positive Rate','Accuracy','Recall','Precision','F1 Score'])

```

(136)

```

plt.plot([SVCtpr,SVCfprA,SVCfprB,SVCfprC,SVCfprD,SVCfprE], 'go-',
[SVCtpr,SVCtprA,SVCtprB,SVCtprC,SVCtprD,SVCtprE], 'ro-',
[acc,accA,accB,accC,accD,accE], 'bo-',
[recall,recallA,recallB,recallC,recallD,recallE], 'co-',
[precision,precisionA,precisionB,precisionC,precisionD,precisionE], 'mo-',
[f1,f1_A,f1_B,f1_C,f1_D,f1_E], 'yo-')

```

(137)

```

plt.title("Results of Support Vector Machine Model(NSL-KDD)")

```

(138)

```

plt.xlabel("False Positive Rate = Green \n True Positive Rate = Red\n
Accuracy Score= Blue\nRecall = Cyan\nPrecision = Magenta\n
F1 Score = Yellow")

```

(139)

```

plt.show()

```

(140)

Next, to train the model with NaBioT dataset.

```

print("-----Wireless Model-----")

```

(140)

```

print("-----DT-----")

```

(141)

```

print("-----Top 10 Features-----")

```

(142)

```
DTtpr, DTfpr, acc, recall,precision,f1=  
ModelDT(bestSelectedFeatureWireless,bestWirelessTest,  
conditionWireless, WirelessExpected) (143)
```

```
print("-----Combination A-----") (144)
```

```
DTtprA, DTfprA, accA, recallA,precisionA,f1_A=  
ModelDT(bestSelectedFeatureWirelessA,bestWirelessTestA,  
conditionWireless, WirelessExpected) (145)
```

```
print("-----Combination B-----") (146)
```

```
DTtprB, DTfprB, accB, recallB,precisionB,f1_B=  
ModelDT(bestSelectedFeatureWirelessB,bestWirelessTestB,  
conditionWireless, WirelessExpected) (147)
```

```
print("-----Combination C-----") (149)
```

```
DTtprC, DTfprC, accC, recall,precisionC,f1_C=  
ModelDT(bestSelectedFeatureWirelessC,bestWirelessTestC,  
conditionWireless, WirelessExpected) (150)
```

```
print("-----Combination D-----") (151)
```

```
DTtprD, DTfprD, accD, recallD,precisionD,f1_D=  
ModelDT(bestSelectedFeatureWirelessD,bestWirelessTestD,  
conditionWireless, WirelessExpected) (152)
```

```
print("-----Combination E-----") (153)
```

```
DTtprE, DTfprE, accE, recallE,precisionE,f1_E=  
ModelDT(bestSelectedFeatureWirelessE,bestWirelessTestE,  
conditionWireless, WirelessExpected) (154)
```

```
data = {'Top 10 Features','Combination A','Combination B',
```



```

'Combination C', 'Combination D', 'Combination E'],
'False Positive Rate':[DTfpr,DTfprA,DTfprB,DTfprC,DTfprD,DTfprE],
'True Positive Rate':[DTtpr,DTtprA,DTtprB,DTtprC,DTtprD,DTtprE],
'Accuracy':[acc,accA,accB,accC,accD,accE],
'Recall':[recall,recallA,recallB,recallC,recallD,recallE],
'Precision':[precision,precisionA,precisionB,
precisionC,precisionD,precisionE],
'F1 Score':[f1,f1_A,f1_B,f1_C,f1_D,f1_E] }

```

(155)

```

df = pd.DataFrame (data, columns = ['False Positive Rate',
'True Positive Rate', 'Accuracy', 'Recall', 'Precision', 'F1 Score'])

```

(156)

```

plt.plot([DTfpr,DTfprA,DTfprB,DTfprC,DTfprD,DTfprE], 'go-',
[DTtpr,DTtprA,DTtprB,DTtprC,DTtprD,DTtprE], 'ro-',
[acc,accA,accB,accC,accD,accE], 'bo-',
[recall,recallA,recallB,recallC,recallD,recallE], 'co-',
[precision,precisionA,precisionB,precisionC,precisionD,precisionE], 'mo-',
[f1,f1_A,f1_B,f1_C,f1_D,f1_E], 'yo-')

```

(157)

```

plt.title("Results of Decision Tree Model(NaBIoT)")

```

(158)

```

plt.xlabel("False Positive Rate = Green \n True Positive Rate = Red\n
Accuracy Score= Blue\nRecall = Cyan\nPrecision = Magenta\n
F1 Score = Yellow")

```

(159)

```

plt.show()

```

(160)

```

print("-----NB-----")

```

(161)

```

print("-----Top 10 Features-----")

```

(162)

```

NBtpr, NBfpr, acc, recall,precision,f1_=

```

*ModelNB(bestSelectedFeatureWireless,bestWirelessTest,  
conditionWireless, WirelessExpected)* (163)

*print("-----Combination A-----")* (164)

*NBtprA, NBfprA, accA, recallA,precisionA,f1\_A=*

*ModelNB(bestSelectedFeatureWirelessA,bestWirelessTestA,  
conditionWireless, WirelessExpected)* (165)

*print("-----Combination B-----")* (166)

*NBtprB, NBfprB, accB, recallB,precisionB,f1\_B=*

*ModelNB(bestSelectedFeatureWirelessB,bestWirelessTestB,  
conditionWireless, WirelessExpected)* (167)

*print("-----Combination C-----")* (168)

*NBtprC, NBfprC, accC, recallC,precisionC,f1\_C=*

*ModelNB(bestSelectedFeatureWirelessC,bestWirelessTestC,  
conditionWireless, WirelessExpected)* (169)

*print("-----Combination D-----")* (170)

*NBtprD, NBfprD, accD, recallD,precisionD,f1\_D=*

*ModelNB(bestSelectedFeatureWirelessD,bestWirelessTestD,  
conditionWireless, WirelessExpected)* (171)

*print("-----Combination E-----")* (172)

*NBtprE, NBfprE, accE, recallE,precisionE,f1\_E=*

*ModelNB(bestSelectedFeatureWirelessE,bestWirelessTestE,  
conditionWireless, WirelessExpected)* (173)

*data = {'Top 10 Features','Combination A','Combination B',  
'Combination C', 'Combination D', 'Combination E'],*

```

'False Positive Rate':[NBtpr,NBfprA,NBfprB,NBfprC,NBfprD,NBfprE],
'True Positive Rate':[NBtpr,NBtprA,NBtprB,NBtprC,NBtprD,NBtprE],
'Accuracy':[acc,accA,accB,accC,accD,accE],
'Recall':[recall,recallA,recallB,recallC,recallD,recallE],
'Precision':[precision,precisionA,precisionB,precisionC,
precisionD,precisionE],
'F1 Score':[f1,f1_A,f1_B,f1_C,f1_D,f1_E] }

```

(174)

```

df = pd.DataFrame (data, columns = ['False Positive Rate',
'True Positive Rate','Accuracy','Recall','Precision','F1 Score'])

```

(175)

```

plt.plot([NBfpr,NBfprA,NBfprB,NBfprC,NBfprD,NBfprE],'go-',
[NBtpr,NBtprA,NBtprB,NBtprC,NBtprD,NBtprE],'ro-',
[acc,accA,accB,accC,accD,accE],'bo-',
[recall,recallA,recallB,recallC,recallD,recallE],'co-',
[precision,precisionA,precisionB,precisionC,
precisionD,precisionE],'mo-',
[f1,f1_A,f1_B,f1_C,f1_D,f1_E],'yo-')

```

(176)

```

plt.title("Results of Naive Bayes Model(NSL-KDD)")

```

(177)

```

plt.xlabel("False Positive Rate = Green \n True Positive Rate = Red\n
Accuracy Score= Blue\nRecall = Cyan\nPrecision = Magenta\n
F1 Score = Yellow")

```

(178)

```

plt.show()

```

(179)

```

print("-----RF-----")

```

(180)

```

print("-----Top 10 Features-----")

```

(181)

```

RFtpr, RFfpr, acc, recall,precision,f1=

```

```

ModelRF(bestSelectedFeatureWireless,bestWirelessTest,

```

*conditionWireless, WirelessExpected)* (182)

*print("-----Combination A-----")* (183)

*RFtprA, RFfprA, accA, recallA,precisionA,f1\_A=*

*ModelRF(bestSelectedFeatureWirelessA,bestWirelessTestA,*  
*conditionWireless, WirelessExpected)* (184)

*print("-----Combination B-----")* (185)

*RFtprB, RFfprB, accB, recallB,precisionB,f1\_B=*

*ModelRF(bestSelectedFeatureWirelessB,bestWirelessTestB,*  
*conditionWireless, WirelessExpected)* (186)

*print("-----Combination C-----")* (187)

*RFtprC, RFfprC, accC, recallC,precisionC,f1\_C=*

*ModelRF(bestSelectedFeatureWirelessC,bestWirelessTestC,*  
*conditionWireless, WirelessExpected)* (188)

*print("-----Combination D-----")* (189)

*RFtprD, RFfprD, accD, recallD,precisionD,f1\_D=*

*ModelRF(bestSelectedFeatureWirelessD,bestWirelessTestD,*  
*conditionWireless, WirelessExpected)* (190)

*print("-----Combination E-----")* (191)

*RFtprE, RFfprE, accE, recallE,precisionE,f1\_E=*

*ModelRF(bestSelectedFeatureWirelessE,bestWirelessTestE,*  
*conditionWireless, WirelessExpected)* (192)

*data = {'Top 10 Features','Combination A','Combination B',*

*'Combination C', 'Combination D', 'Combination E'],*

*'False Positive Rate':[RFtpr,RFfprA,RFfprB,RFfprC,RFfprD,RFfprE],*

```

'True Positive Rate':[RFtpr,RFtprA,RFtprB,RFtprC,RFtprD,RFtprE],
'Accuracy':[acc,accA,accB,accC,accD,accE],
'Recall':[recall,recallA,recallB,recallC,recallD,recallE],
'Precision':[precision,precisionA,precisionB,precisionC,
precisionD,precisionE],
'F1 Score':[f1,f1_A,f1_B,f1_C,f1_D,f1_E]

```

(193)

```

df = pd.DataFrame (data, columns = ['','False Positive Rate',
'True Positive Rate','Accuracy','Recall','Precision','F1 Score'])

```

(194)

```

plt.plot([RFtpr,RFfprA,RFfprB,RFfprC,RFfprD,RFfprE], 'go-',
[RFtpr,RFfprA,RFfprB,RFfprC,RFfprD,RFfprE], 'ro-',
[acc,accA,accB,accC,accD,accE], 'bo-',
[recall,recallA,recallB,recallC,recallD,recallE], 'co-',
[precision,precisionA,precisionB,precisionC,precisionD,precisionE], 'mo-',
[f1,f1_A,f1_B,f1_C,f1_D,f1_E], 'yo-')

```

(195)

```

plt.title("Results of Random Forest Model(NSL-KDD)")

```

(196)

```

plt.xlabel("False Positive Rate = Green \n True Positive Rate = Red\n
Accuracy Score= Blue\nRecall = Cyan\nPrecision = Magenta\n
F1 Score = Yellow")

```

(197)

```

plt.show()

```

(198)

```

print("-----SVC-----")

```

(199)

```

print("-----Top 10 Features-----")

```

(200)

```

SVCtpr, SVCfpr, acc, recall,precision,f1=
ModelSVC(bestSelectedFeatureWireless,bestWirelessTest,
conditionWireless, WirelessExpected)

```

(201)

```

print("-----Combination A-----")

```

(202)

```
SVCtprA, SVCfprA, accA, recallA,precisionA,f1_A=
ModelSVC(bestSelectedFeatureWirelessA,bestWirelessTestA,
conditionWireless, WirelessExpected) (203)
```

```
print("-----Combination B-----") (204)
```

```
SVCtprB, SVCfprB, accB, recallB,precisionB,f1_B=
ModelSVC(bestSelectedFeatureWirelessB,bestWirelessTestB,
conditionWireless, WirelessExpected) (205)
```

```
print("-----Combination C-----") (206)
```

```
SVCtpr, SVCfprC, accC, recallC,precisionC,f1_C=
ModelSVC(bestSelectedFeatureWirelessC,bestWirelessTestC,
conditionWireless, WirelessExpected) (207)
```

```
print("-----Combination D-----") (208)
```

```
SVCtprD, SVCfprD, accD, recallD,precisionD,f1_D=
ModelSVC(bestSelectedFeatureWirelessD,bestWirelessTestD,
conditionWireless, WirelessExpected) (210)
```

```
print("-----Combination E-----") (211)
```

```
SVCtprE, SVCfprE, accE, recallE,precisionE,f1_E=
ModelSVC(bestSelectedFeatureWirelessE,bestWirelessTestE,
conditionWireless, WirelessExpected) (212)
```

```
data = {'':['Top 10 Features','Combination A','Combination B',
'Combination C', 'Combination D', 'Combination E'],
'False Positive Rate':[SVCtpr,SVCfprA,SVCfprB,
SVCfprC,SVCfprD,SVCfprE],
'True Positive Rate':[SVCtpr,SVCtprA,SVCtprB,
```

```

SVCtprC,SVCtprD,SVCtprE],
'Accuracy':[acc,accA,accB,accC,accD,accE],
'Recall':[recall,recallA,recallB,recallC,recallD,recallE],
'Precision':[precision,precisionA,precisionB,precisionC,
precisionD,precisionE],
'F1 Score':[f1,f1_A,f1_B,f1_C,f1_D,f1_E]}

```

(213)

```

df = pd.DataFrame (data, columns = ['False Positive Rate',
'True Positive Rate','Accuracy','Recall','Precision','F1 Score'])

```

(214)

```

plt.plot([SVCtpr,SVCfprA,SVCfprB,SVCfprC,SVCfprD,SVCfprE], 'go-',
[SVCtpr,SVCtprA,SVCtprB,SVCtprC,SVCtprD,SVCtprE], 'ro-',
[acc,accA,accB,accC,accD,accE], 'bo-',
[recall,recallA,recallB,recallC,recallD,recallE], 'co-',
[precision,precisionA,precisionB,precisionC,precisionD,precisionE], 'mo-',
[f1,f1_A,f1_B,f1_C,f1_D,f1_E], 'yo-')

```

(215)

```

plt.title("Results of Support Vector Machine Model(NSL-KDD)")

```

(216)

```

plt.xlabel("False Positive Rate = Green \n True Positive Rate = Red\n
Accuracy Score= Blue\nRecall = Cyan\nPrecision = Magenta\n
F1 Score = Yellow")

```

(217)

```

plt.show()

```

(218)

## Chapter 4: Flow diagram

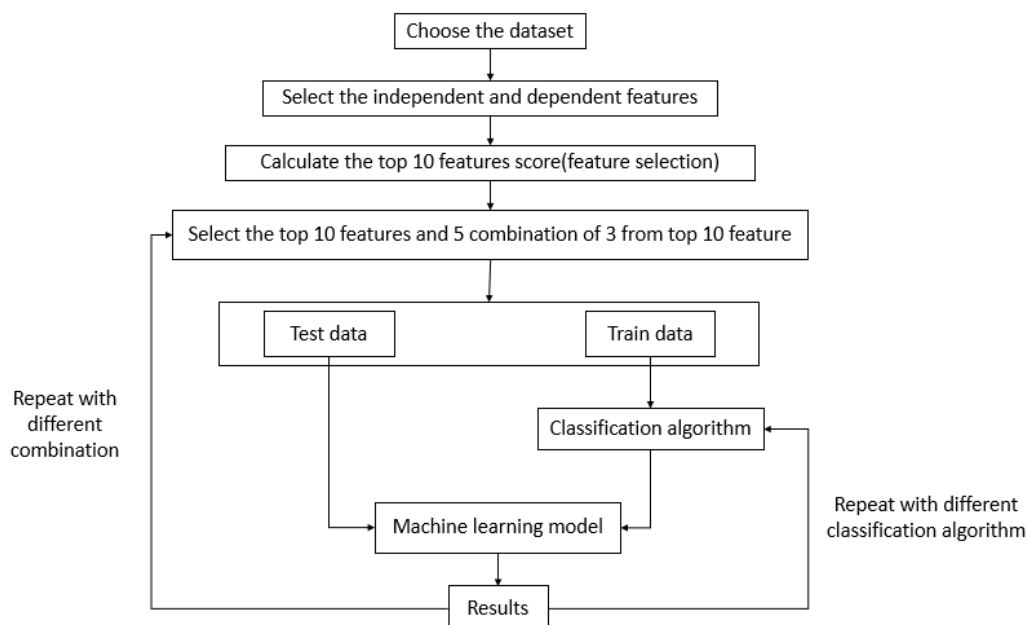


Figure 4(a) : Work flow of this research

In this research, there are two type of dataset used to train the model which is wired and wireless. The wired dataset which has 41 features was used to train the model is KDDTrain+ with testing dataset KDDTest+. Besides, for wireless dataset its from NBIoT dataset from Y. Meidan et. al. consists of 115 features. For NSL-KDD dataset the top 10 features were calculated by using ANOVA. The top 10 features were used to train the model and the precision, recall, f score and accuracy was used to determine the performance of the model. Besides, 10 groups of random combination of top 10 features are used to train the model to get the accuracy score. Each of the group consist of 3 features from top 10 features. NBIoT dataset consist of two attack which is Mirai attack and Bashlite attack which also known as Gafgyt, Q-Bot, Torlus, LizardStresser, and Lizkebab. Both of this attack is Distributed Denial of Service(DDoS) attack which use Internet of Things(IoT) device as botnet to perform this attack to a target. It is controlled by a bot master to launch an attack. There are 5 different files in Mirai attack which is scan, ACK, SYN, UDP and UDP plain. These 5 files are the attack the launch by Mirai attack. Scan is the file which consist of the packet the Mirai attack perform scanning for vulnerable device in the network. In addition, ACK, SYN, UDP and UDP plain consist of the packet that used to flood a target device. Bashlite attack is also one of the attack in IoT, which is also performing DDoS attack by using infected botnet. In Bashlite attack it consist of 5 files which is scan, junk, UDP, TCP and combo. Same as



Mirai attack, Bashlite attack also consist of scanning vulnerable device in the network. UDP and TCP consists of the packet of flooding the target via UDP and TCP while junk consist of the packet which the device send junk data to the target. Combo consist of the packet that send spam data and open a connection to a specific IP and port. The wireless train dataset was created by combining the 5 files in Mirai attack and normal traffic which is the benign traffic which provided in the NaBIoT dataset. For the wireless test dataset, it also consists of 5 files in Mirai attack and benign traffic. On top of that the wireless test dataset was also combined with 5 files in Bashlite attack.

The next step is to select the independent feature and dependent feature of the dataset before feature selection. The independent feature act as the input for the model and dependent feature act as the output for the independent feature. In NSL-KDD, all of the feature was selected as independent feature to train the model except for the label of the dataset which is the type of attacks or normal. The label of the dataset act as the dependent feature for NSL-KDD this is due to the dependent feature is depend on the independent feature to produce the output which is what is labelled on the dataset.

The third step of this work flow is to have feature selection of the independent feature. The algorithm that used to calculate the feature score is `SelectKBest(score_func=f_classif, k=10)`. The function will calculate the top 10 feature scores and the parameter used as the score function is ANOVA (`f_classif`) and the parameter `k` is the number of feature that are needed as result after the calculation of the feature scores. In this research top 10 feature are needed, therefore, the `k` value will be 10. The relevant features regarding a set of class labels was measure by ANOVA and it is used for feature selection in various fields. There are some works that shows ANOVA is a powerful statistic test. It can be used to rank individual feature relevance in supervised classification. (M. Peña et al, 2017). The top 10 feature score was calculated and the order of the feature score starts from the highest to the lowest. In figure , the feature score which has the highest score is `same_srv_rate` which has a score of  $1.64e+05$  for NSL-KDD and the feature that has the lowest score is `count` which has the score of  $6.27e+04$ . Besides, the top 10 scores for NaBIoT dataset was also calculate by `SelectKBest(score_func=f_classif, k=10)`. Same as NSL-KDD feature scoring, the function used is `SelectKBest(score_func=f_classif, k=10)`, the parameter used in this function is ANOVA which is the algorithm to calculate the top 10 scores and `k=10` which is the top 10 feature scores. In figure , the feature that has the highest scoring in

NaBIoT is MI\_dir\_L0.1\_weight with the score of 1.52e+04 and the lowest score in the top 10 is H\_L5\_weight which has a scoring of 2.18e+03.

Top 10 Feature	Score
same_srv_rate	1.64e+05
dst_host_srv_count	1.38e+05
dst_host_same_srv_rate	1.17e+05
logged_in	1.15e+05
dst_host_srv_serror_rate	9.46e+04
dst_host_serror_rate	9.31e+04
serror_rate	9.25e+04
srv_serror_rate	9.13e+04
Flag	7.48e+04
count	6.27e+04

Figure 4(b): Top 10 feature scores for NSL-KDD dataset

Top 10 Feature	Score
MI_dir_L0.1_weight	1.52e+04
H_L0.1_weight	1.52e+04
MI_dir_L1_weight	6.01e+03
H_L1_weight	6.01e+03
MI_dir_L0.01_weight	3.11e+03
H_L0.01_weight	3.11e+03
MI_dir_L3_weight	2.89e+03
H_L3_weight	2.89e+03
MI_dir_L5_weight	2.18e+03
H_L5_weight	2.18e+03

Figure 4(c): Top 10 feature scores for NaBIoT dataset

The fourth step for this work flow will be selecting the independent feature after feature scoring. After feature scoring for independent feature, the top 10 features were

selected to fit into the model in order to train the model. Besides, 5 combinations of 3 from the feature was selected to train the model.

Combination A	<ol style="list-style-type: none"> <li>1. Same_srv_rate, (the percentage of connections that were to the same service among the connections aggregated in count)</li> <li>2. Dst_host_srv_count,(number of connections having the same port number)</li> <li>3. Dst_host_same_srv_rate(the percentage of connections that were to the same service among the connection aggregated in dst_host_count)</li> </ol>
Combination B	<ol style="list-style-type: none"> <li>1. Count (number of connections to the same destination host as the current connection in the past)</li> <li>2. Logged_in (log in status 1 for successfully logged in, 0 for unsuccessful)</li> <li>3. Dst_host_same_srv_rate(the percentage of connections that were to the same service among the connections aggregated in dst_host_count)</li> </ol>
Combination C	<ol style="list-style-type: none"> <li>1. Serror_rate (the percentage of onnections that have activated the flag s0, s1, s2 or s3 among the connections aggregated in count.)</li> <li>2. Srv_error_rate (the percentage of connections that have activated the flag s0, s1, s2 or s3 amount the connections aggregated in srv_count.)</li> <li>3. Flag (status of the connection, it is either normal or error)</li> </ol>
Combination D	<ol style="list-style-type: none"> <li>1. Srv_error_rate (the percentage of connections that have activated the flag s0, s1, s2 or s3 amount the connections aggregated in srv_count.)</li> <li>2. Same_srv_rate (the percentage of connections that were to the same service among the connections aggregated in count.)</li> <li>3. Dst_host_same_srv_rate (the percentage of connections that were to the same service among the connections aggregated in dst_host_count)</li> </ol>
Combination E	<ol style="list-style-type: none"> <li>1. Flag (status of the connection, it is either normal or error)</li> <li>2. Dst_host_srv_count (number of connections having the same port number.)</li> <li>3. Dst_host_srv_error_rate (the percent of connections that have activated flag s0,s1,s2 or s3 among the connections aggregated in dst_host_srv_count.)</li> </ol>

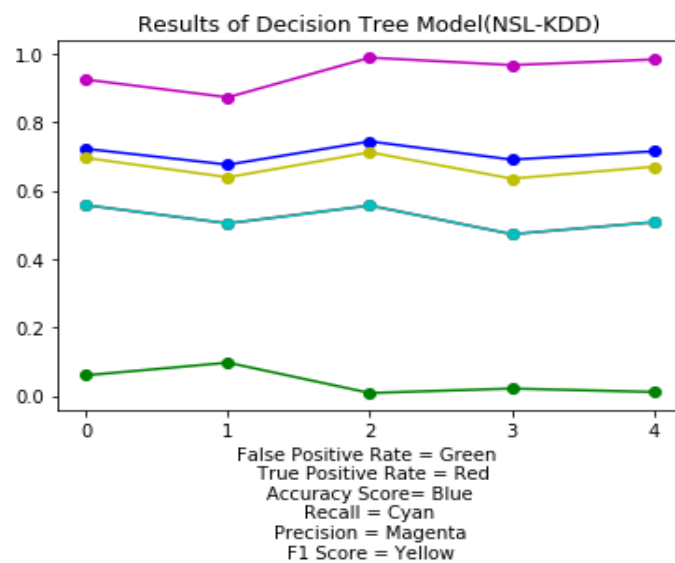
Table 4(a): Different Combinations of Features

Combination A	<ol style="list-style-type: none"> <li>1. MI_dir_L0.1_weight (The weight of the stream of the summarize packet from the following host's packet (IP+MAC) in the time frame L0.1)</li> <li>2. H_L0.1_weight (The weight of the stream of the summarize packet form the following host's packet (IP) in the time frame L0.1)</li> <li>3. MI_dir_L1_weight (The weight of the stream of the summarize packet from the following host's packet (IP+MAC) in the time frame L1)</li> </ol>
Combination B	<ol style="list-style-type: none"> <li>1. H_L1_weight (The weight of the stream of the summarize packet form the following host's packet (IP) in the time frame L1)</li> <li>2. MI_dir_L0.01_weight (The weight of the stream of the summarize packet from the following host's packet (IP+MAC) in the time frame L0.01)</li> <li>3. H_L0.01_weight(The weight of the stream of the summarize packet form the following host's packet (IP) in the time frame L0.01)</li> </ol>
Combination C	<ol style="list-style-type: none"> <li>1. MI_dir_L3_weight (The weight of the stream of the summarize packet from the following host's packet (IP+MAC) in the time frame L3)</li> <li>2. H_L3_weight (The weight of the stream of the summarize packet form the following host's packet (IP) in the time frame L3)</li> <li>3. MI_dir_L5_weight (The weight of the stream of the summarize packet from the following host's packet (IP+MAC) in the time frame L5)</li> </ol>
Combination D	<ol style="list-style-type: none"> <li>1. MI_dir_L0.1_weight (The weight of the stream of the summarize packet from the following host's packet (IP+MAC) in the time frame L0.1)</li> <li>2. H_L0.01_weight (The weight of the stream of the summarize packet form the following host's packet (IP) in the time frame L0.01)</li> <li>3. H_L5_weight (The weight of the stream of the summarize packet form the following host's packet (IP) in the time frame L5)</li> </ol>
Combination E	<ol style="list-style-type: none"> <li>1. H_L5_weight (The weight of the stream of the summarize packet form the following host's packet (IP) in the time frame L5)</li> <li>2. H_L1_weight (The weight of the stream of the summarize packet form the following host's packet (IP) in the time frame L1)</li> <li>3. H_L0.01_weight (The weight of the stream of the summarize packet form the following host's packet (IP) in the time frame L0.01)</li> </ol>

Table 4 (b): Combination for NaBIoT dataset

After the selection for the combination, the train data and test data was selected according to the combination in table 4(a) and table 4(b). NSL-KDD dataset was used to train decision tree mode, naïve bayes mode, random forest model, and support vector machine model. The legends for combination A (0), combination B (1), combination C (2), combination D (3) and combination E (4). The results of decision tree classifier were shown in figure 4(d). The model uses test data to make prediction and also calculate the outcome of the model. The model is fitted combination A to train the model. The false positive rate is 0.0601 and true positive rate is 0.557 for combination A. The accuracy of the model is 0.722, recall is 0.557, precision is 0.925 and f1 score is 0.695. The second combination which is combination B was used to train the model.

The false positive rate is 0.0969 and true positive rate is 0.504. The accuracy of the model is 0.676, recall is 0.504, precision is 0.873 and f1 score is 0.639. Next, the third combination which is combination C was used to train the model. The false positive rate of the trained model is 0.00814 and true positive rate is 0.556. The accuracy of this model is 0.744, recall is 0.556, precision is 0.989 and f1 score is 0.712. The fourth combination used to train the model is combination D. The false positive rate is 0.0212 and true positive rate is 0.473. The accuracy of this model is 0.691, recall is 0.473, precision is 0.973 and f1 score is 0.635. The final combination used to train the model is combination E. The false positive rate of the model is 0.0111 and true positive rate is 0.473. The accuracy of this model is 0.691, recall is 0.473, precision is 0.967 and f1 score is 0.635. Among all of the combination, the combination that has the lowest false positive rate, highest true positive rate and highest accuracy is combination C. Out of all of the combination, combination C gives the most accurate prediction with low false positive rate.



	False Positive Rate	True Positive Rate	Accuracy	Recall	Precision	F1 Score
Combination A	0.0601	0.557	0.722	0.557	0.925	0.696
Combination B	0.0969	0.504	0.676	0.504	0.873	0.639
Combination C	0.00814	0.556	0.744	0.556	0.989	0.712
Combination D	0.0212	0.473	0.691	0.473	0.967	0.635
Combination E	0.0111	0.51	0.716	0.51	0.984	0.672

Figure 4(d) : Results for Decision Tree Model (NSL-KDD)

Next, naïve bayes was used to train the same combinations from NSL-KDD dataset. The legends for combination A (0), combination B (1), combination C (2), combination D (3) and combination E (4). The results of naïve bayes classifier is shown in figure 4(e). Combination A was used to train the model. The results that achieve from the model is false positive rate which is 0.0786, true positive rate is 0.579, accuracy is 0.727, recall is 0.579, precision is 0.907 and f1 score is 0.707. The next combination used to train the model is combination B. The false positive rate of the model is 0.0829 and true positive rate is 0.488. The accuracy for this model is 0.673, recall is 0.488, precision is 0.886 and f1 score is 0.630. After that, the combination C was used to train the model. The false positive rate from combination C is 0.00536 and true positive rate 0.256. The accuracy of this combination is 0.574, recall is 0.256, precision is 0.984 and f1 score is 0.406. The fourth combination that used to train the model is combination D. The false positive rate of the model is 0.0233 and true positive rate is 0.546. The accuracy of the model is 0.732, recall is 0.546, precision is 0.969 and f1 score is 0.699. The final combination that used to train the model is combination E. The false positive rate of the model is 0.0136 and true positive rate is 0.563. The accuracy of this model is 0.745, recall is 0.563, precision is 0.982 and f1 score is 0.716. Among of all the combination, combination E gives the best results. Combination C does gives the lowest false positive rate but the accuracy for combination C is only 0.574 while combination E has 0.745. For top 10 feature, it does give the highest true positive rate. But the accuracy of top 10 feature is rather low as compare to combination E. Therefore, combination E gives the best result in false positive rate, true positive rate, accuracy, recall, precision and f1 score.

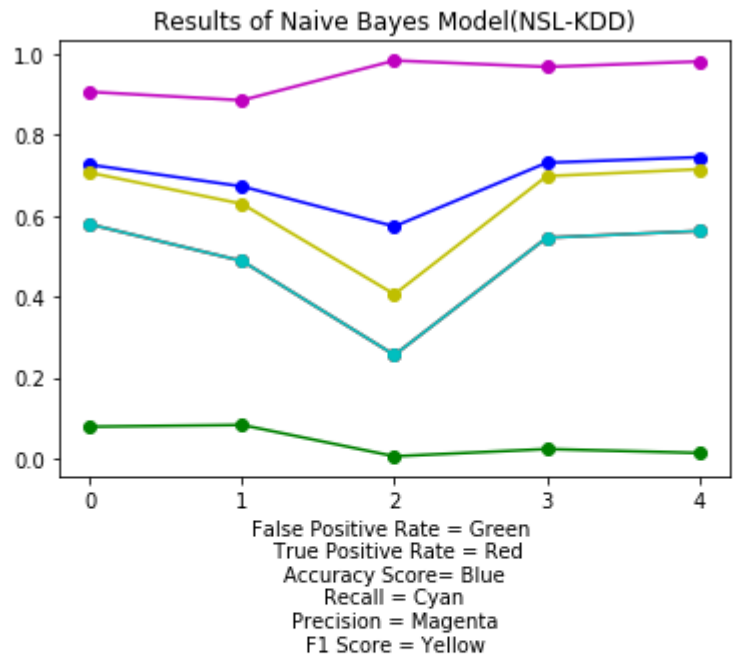
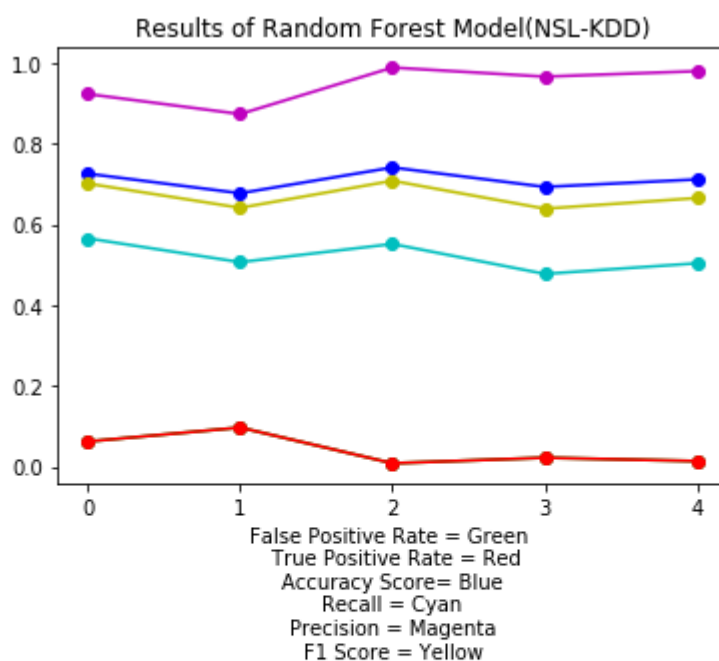


Figure 4(e) : Results for Naïve Bayes Model(NSL-KDD)

The third machine learning algorithm is random forest classifier. The legends for combination A (0), combination B (1), combination C (2), combination D (3) and combination E (4). The results for random forest classifier is shown in figure 4(f). The first combination used to train the model is combination A. The false positive rate for this model is 0.0621 and true positive rate is 0.566. The accuracy of this model is 0.726, recall is 0.566, precision is 0.923 and f1 score is 0.702. The second combination used to train the model is combination B. It has the false positive rate of 0.0968 and true positive rate of 0.506. The accuracy for this combination is 0.677, recall is 0.507, precision is 0.874 and f1 score is 0.641. The third combination used to train the model is combination C. The false positive rate is 0.00814 and true positive rate is 0.552. The accuracy of this combination is 0.741, recall is 0.552, precision is 0.989 and f1 score is 0.708. The fourth combination used to train the model is combination D. It has the false positive rate of 0.0219 and true positive rate of 0.478. The accuracy for this model is

0.693, recall is 0.478, precision is 0.966 and f1 score is 0.639. The final combination used to train the model is combination E. The false positive rate of this model is 0.0136 and true positive rate is 0.504. The accuracy of this model is 0.712, recall is 0.504, precision is 0.980 and f1 score is 0.666. Among all of the combination, combination C gives the best result. It has the lowest false positive rate 0.00814, true positive rate of 0.552 and accuracy of 0.741. Although, combination A has the highest true positive rate but other's score of combination A is lower than combination C. Therefore, the best combination that can conclude in random forest is combination C.



	False Positive Rate	True Positive Rate	Accuracy	Recall	Precision	F1 Score
Combination A	0.0621	0.566	0.726	0.566	0.923	0.702
Combination B	0.0968	0.507	0.677	0.507	0.874	0.641
Combination C	0.00814	0.552	0.741	0.552	0.989	0.708
Combination D	0.0219	0.478	0.693	0.478	0.966	0.639
Combination E	0.0136	0.504	0.712	0.504	0.98	0.666

Figure 4(f): Results for Random Forest Model (NSL-KDD)

The final classifier that used NSL-KDD dataset to train the model is support vector machine. The legends for combination A (0), combination B (1), combination C (2), combination D (3) and combination E (4). The results of support vector machine classifier is shown in figure 4(g). The first combination that used to train the model is



combination A. The false positive rate of the model is 0.0375 and the true positive rate is 0.518. The accuracy for this combination is 0.709, recall is 0.518, precision is 0.948 and f1 score is 0.670. The second combination that used to train the model is combination B. It has the false positive rate of 0.0994 and true positive rate of 0.504. It has the accuracy of 0.675, recall is 0.504, precision is 0.870 and f1 score is 0.638. The fourth combination that used to train the model is combination C. This combination gives a false positive rate of 0.00834 and true positive rate of 0.563. The accuracy of this model is 0.745, recall is 0.563, precision is 0.989 and f1 score is 0.718. The fifth combination used to train the model is combination D. The false positive rate of this model is 0.0141 and true positive rate of 0.479. The accuracy of this combination is 0.697, recall is 0.479, precision is 0.978 and f1 score is 0.643. The final combination that used to train the model is combination E. It has the false positive rate of 0.00855 and true positive rate of 0.492. The accuracy of this combination is 0.707, recall is 0.492, precision is 0.987 and f1 score is 0.657. Among all of the combination, combination C gives the best results. It has the lowest false positive rate, highest true positive rate and highest accuracy as compare to the other combinations.

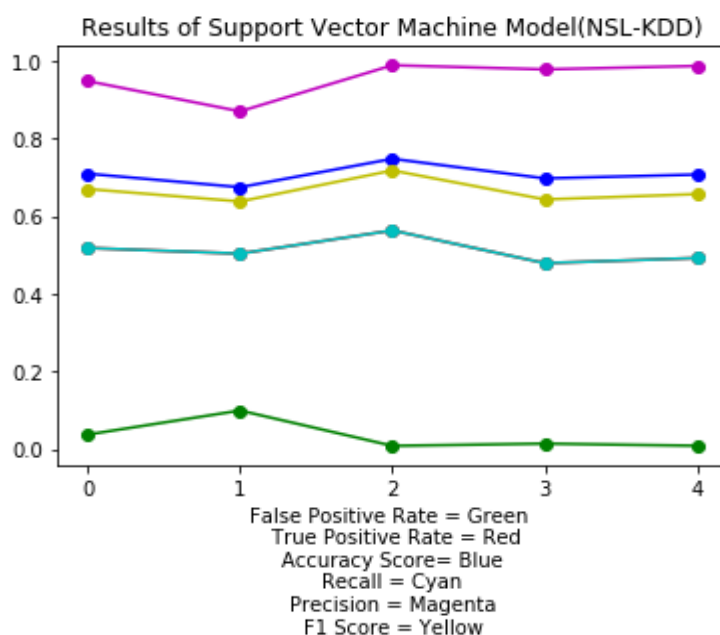
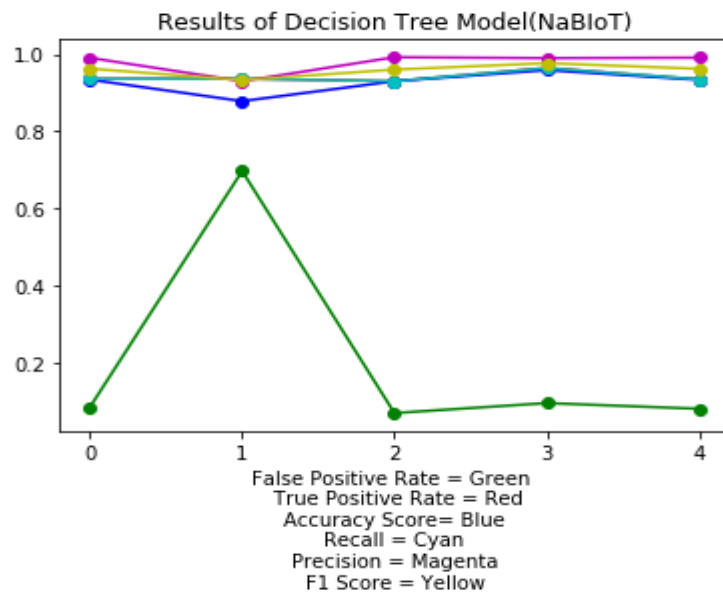


Figure 4(g): Results for Support Vector Machine Model (NSL-KDD)

All of the combination of NSL-KDD datasets is tested with different machine learning algorithm and the results are gathered. The same process is repeated with NaBIoT dataset. The legends for combination A (0), combination B (1), combination C (2), combination D (3) and combination E (4). The results of decision tree classifier for NaBIoT is shown in figure 4(h). The first combination is combination A. The false positive rate for this combination is 0.0849 and true positive rate is 0.936. The accuracy for this combination is 0.934, recall is 0.936, precision is 0.991 and f1 score is 0.963. The combination B was used as the second combination to train the model. It has false positive rate of 0.697, true positive rate of 0.937, accuracy of 0.879, recall of 0.418, precision of 0.869 and f1 score of 0.565. The third combination used to train the model is combination C. The false positive rate of this combination is 0.0701 and true positive rate is 0.93. The accuracy for this combination is 0.93, recall is 0.897, precision is 0.992 and f1 score is 0.96. The fourth combination that used to train the model is combination D. This combination gives a false positive rate of 0.0959, true positive rate of 0.965,

accuracy of 0.959, recall of 0.965, precision of 0.99 and f1 score of 0.997. The final combination that used to train the model is combination E. The false positive rate for this combination is 0.841 and true positive rate is 0.0302. The accuracy is 0.0419, recall is 0.0302, precision is 0.262 and f1 score is 0.0541. Among all of the combination, combination D gives the best results. It has the lowest false positive rate, highest true positive rate and high accuracy which is 0.959.

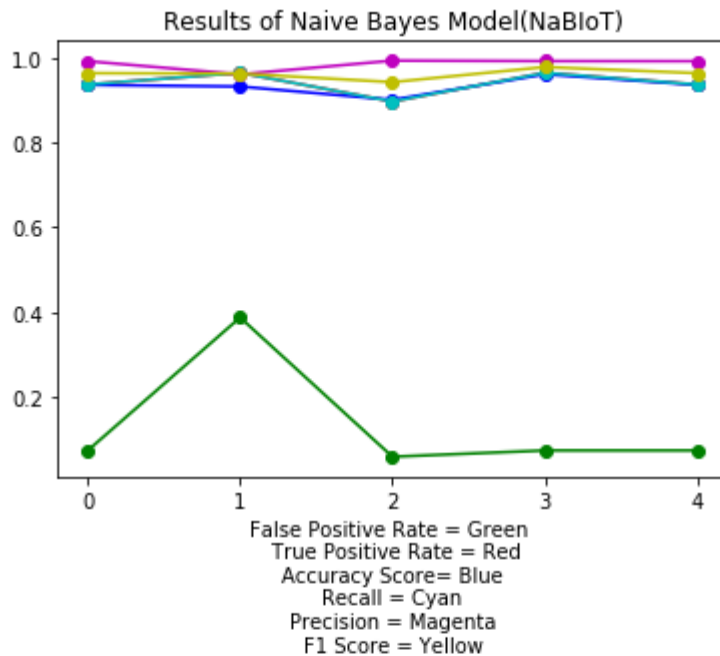


	False Positive Rate	True Positive Rate	Accuracy	Recall	Precision	F1 Score
Combination A	0.0849	0.936	0.934	0.936	0.991	0.963
Combination B	0.697	0.937	0.879	0.937	0.93	0.933
Combination C	0.0701	0.93	0.93	0.93	0.992	0.96
Combination D	0.0959	0.965	0.959	0.965	0.99	0.977
Combination E	0.0812	0.935	0.934	0.935	0.991	0.962

Figure 4(h): Results for Decision Tree Model(NaBIoT)

The next machine learning algorithm used to train the combinations is naïve bayes classifier. The legends for combination A (0), combination B (1), combination C (2), combination D (3) and combination E (4). The results for naïve bayes classifier for NaBIoT is shown in figure 4(i). The first combination that used to train the model is combination A. The false positive rate for this combination is 0.0738 and true positive rate is 0.937. The accuracy for this combination is 0.936, recall is 0.937, precision is 0.992 and f1 score is 0.964. Combination B was used as the second combination that used to train the model. The false positive rate is 0.387 and true positive rate is 0.965.

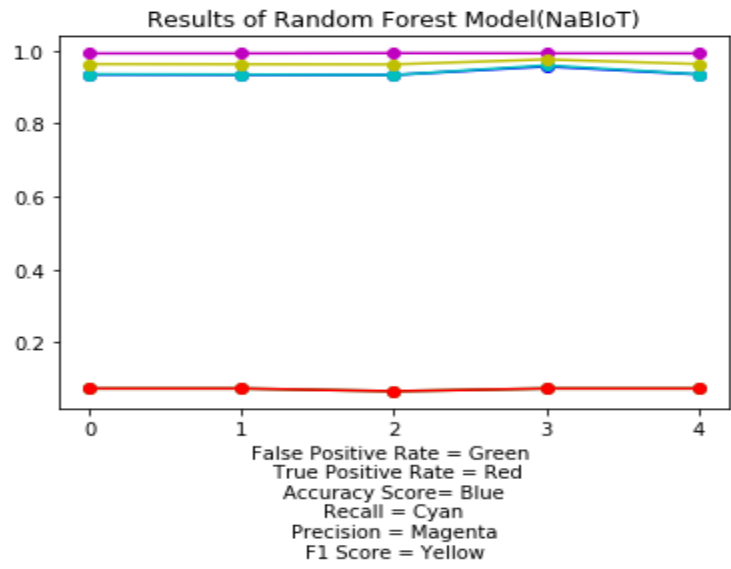
The accuracy of this model is 0.932, recall is 0.965, precision is 0.993 and f1 score is 0.963. Combination C is the third combination that used to train the model. It has the false positive rate of 0.059, true positive rate of 0.897, accuracy of 0.901, recall of 0.897, precision of 0.993 and f1 score of 0.963. The fourth combination that used to train the model is combination D. The false positive rate of this combination is 0.0738, true positive rate of 0.965, accuracy is 0.961, recall is 0.965, precision is 0.992 and f1 score is 0.978. Combination E is the final combination that used to train naïve bayes classifier. It has the false positive rate of 0.395 and true positive rate of 0.965. The accuracy for this combination is 0.932, recall is 0.965, precision is 0.96 and f1 score is 0.962. Among all of the combination, combination D gives the best results. It has the lowest false positive rate and highest accuracy.



	False Positive Rate	True Positive Rate	Accuracy	Recall	Precision	F1 Score
Combination A	0.0738	0.937	0.936	0.937	0.992	0.964
Combination B	0.387	0.965	0.932	0.965	0.961	0.963
Combination C	0.059	0.897	0.901	0.897	0.993	0.943
Combination D	0.0738	0.965	0.961	0.965	0.992	0.978
Combination E	0.0738	0.937	0.936	0.937	0.992	0.964

Figure 4(i): Results for Naïve Bayes Classifier (NaBioT)

The third algorithm used is random forest classifier. The legends for combination A (0), combination B (1), combination C (2), combination D (3) and combination E (4). Combination A is the first combination that used to train the model. False positive rate of this combination is 0.0738 and true positive rate is 0.935. This combination has an accuracy of 0.934, recall of 0.935, precision of 0.992 and f1 score of 0.963. The second combination that used to train the model is combination B. The false positive rate of this combination is 0.0738, true positive rate is 0.934. The accuracy for this model is 0.934, recall is 0.934, precision is 0.992 and f1 score is 0.962. Combination C is the third combination that used to train the model. The false positive rate for this combination is 0.0664 and true positive rate is 0.933. The accuracy for combination C is 0.933, recall is 0.933, precision is 0.992 and f1 score is 0.962. The fourth combination that used to train the model is combination D. The false positive rate for this combination is 0.0738 and true positive rate is 0.959. The accuracy for combination D is 0.956, recall is 0.959, precision is 0.992 and f1 score is 0.976. The last combination is combination E. It has false positive rate of 0.0738 and true positive rate of 0.935. The accuracy for combination E is 0.934, recall is 0.935, precision is 0.992 and f1 score is 0.963. Among all of the combination, combination D has the best results in accuracy, true positive rate, recall, precision and f1 score.

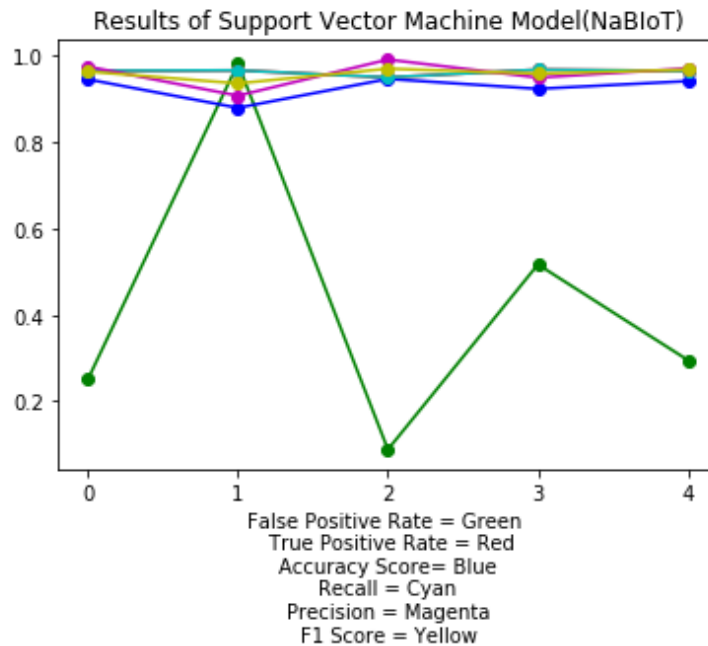


	False Positive Rate	True Positive Rate	Accuracy	Recall	Precision	F1 Score
Combination A	0.0738	0.935	0.934	0.935	0.992	0.963
Combination B	0.0738	0.934	0.934	0.934	0.992	0.962
Combination C	0.0664	0.933	0.933	0.933	0.993	0.962
Combination D	0.0738	0.959	0.956	0.959	0.992	0.976
Combination E	0.0738	0.935	0.934	0.935	0.992	0.963

Figure 4(j): Results for Random Forest Model (NaBioT)

The last machine learning algorithm that used to train the model is support vector machine. The legends for combination A (0), combination B (1), combination C (2), combination D (3) and combination E (4). The results for support vector machine model for NaBioT is shown in figure 4(k). Combination A is the first combination that used to train the model. This combination gives a false positive rate of 0.251 and true positive rate of 0.965. The accuracy for combination A is 0.945, recall is 0.965, precision is 0.974 and f1 score is 0.964. Combination B is the second combination that used to train the model. The false positive rate for combination B is 0.982 and true positive rate is 0.966. The accuracy for this combination is 0.879, recall 0.966, precision is 0.907 and f1 score is 0.935. Next, combination C was used to train the model. It has a false positive rate of 0.0886 and true positive rate of 0.563. The accuracy for combination C is 0.946, recall is 0.949, precision is 0.991 and f1 score is 0.97. Combination D is the fourth combination that used to train the model. The false positive rate for combination D is 0.517 and true positive rate is 0.967. The accuracy for combination D is 0.923, recall is 0.967, precision is 0.949 and f1 score is 0.958. The

last combination that used to train the model is combination E. The false positive rate for this combination is 0.982 and true positive rate is 0.966. The accuracy for combination E is 0.879, recall is 0.966, precision is 0.907 and f1 score is 0.936. Among all of the combination, the combination that gives the best result is combination C. It has the lower false positive rate and highest accuracy among all of the other combination.



	False Positive Rate	True Positive Rate	Accuracy	Recall	Precision	F1 Score
Combination A	0.251	0.965	0.945	0.965	0.974	0.963
Combination B	0.982	0.966	0.879	0.966	0.907	0.935
Combination C	0.0886	0.949	0.946	0.949	0.991	0.97
Combination D	0.517	0.967	0.923	0.967	0.949	0.958
Combination E	0.295	0.965	0.941	0.965	0.97	0.967

Figure 4(k): Results for Support Vector Machine Model (NaBIoT)

## Chapter 5: Discussion

In this research, the input for the model is numerical input and the output of the model will be categorical output. Therefore, ANOVA is the feature selection algorithm that used to measure the top 10 feature score in this research. ANOVA is an algorithm that is based on f-test that has a f distribution under the null hypothesis. The existence variance among numerous population means can be determined by ANOVA which is a statistical technique. It is used to differentiate whether it has difference in two or more datasets are statistically important. (Z.A. Bakar et. al.). The model uses numerical variables as the input which is either integer variables or floating point variables and the output of the model is categorical variables which is Boolean variables. The output of the model is either 1 or 0 which is attack or normal. There are several advantage that are able to achieve in feature selection. It reduces overfitting to the model. In the other words, there is lesser chance that when the model is making decision it is based on the redundant data or noise. Besides, feature selection also increases the accuracy of the model if it is able to find the optimum number of feature to fit into the model. This is due to there is less noise in the data which have the possibility that it will mislead the prediction. The last advantage that is able to obtain by feature selection is reduce training time. For example, in this research the machine learning algorithm that takes the longest time is support vector machine. If all of the feature is fitted into the machine learning model, it will be taking too long to train the model. After feature selection is done, the training time for support vector machine decreases as compare to all of the feature fit into the model. The combination of 3 is randomly selected from the top 10 feature of the dataset.

After all of the training of model has done for both datasets. A comparison between NSL-KDD and NaBIoT was conducted. First, the accuracy between different combinations of features for NSL-KDD was compared. Figure 5(a) shows the accuracy of different combination of features for NSL-KDD dataset. The highest accuracy is combination C (shows in table 4(a)) from support vector machine classifier. The algorithm naïve bayes has the highest accuracy among all of the combinations and classifier, it has an accuracy of 0.748. The combination contains 3 important features which is `error_rate`, `srv_error_rate` and `flag`. `error_rate` is the percentage of connections that activated the flag `s0`, `s1`, `s2` and `s3` among the connection in count.



Srv\_error\_rate is the percentage of connections that activated the flag s0, s1, s2 and s3 among the connections in srv\_count. Flag is the connection status, it is either normal or error. The combination that has the lowest detection accuracy is combination C from naïve bayes. It has only detection accuracy of 0.574.

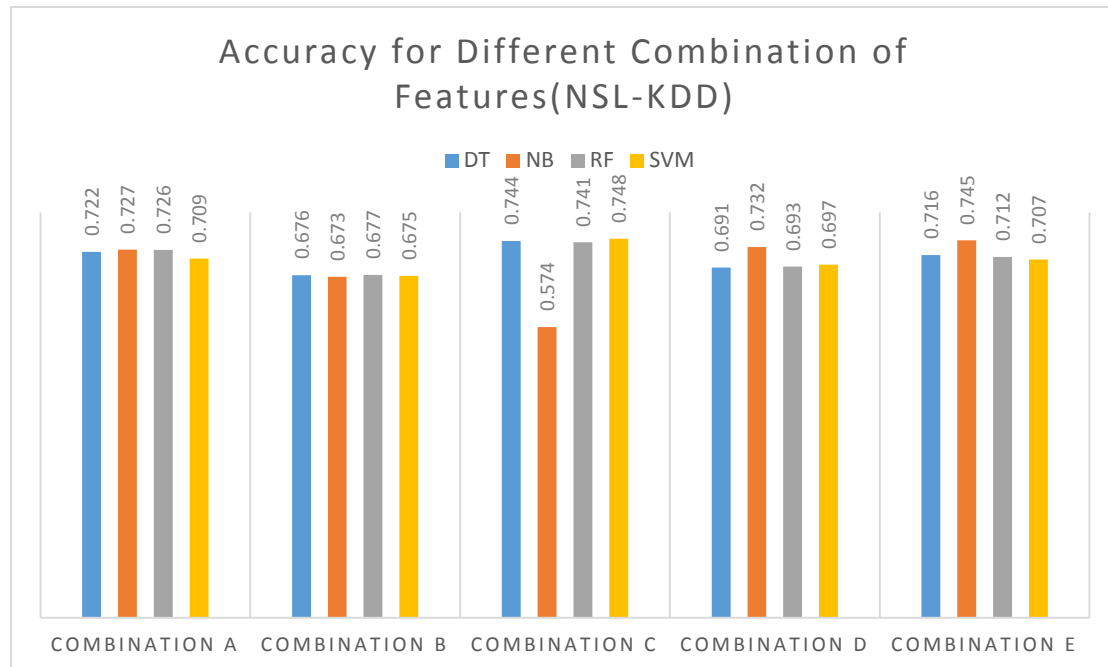


Figure 5(a): Accuracy of Different Combination of Features (NSL-KDD)

In figure 5(b), it shows the accuracy of different combination of features for NaBIoT dataset. The highest accuracy in NaBIoT dataset is 0.961 which is combination D from naïve bayes classifier. The combination has 3 important features that contribute in such high accuracy. The involved features is MI\_dir\_L0.1\_weight, H\_L0.01\_weight and H\_L5\_weight. MI\_dir\_L0.1\_weight is the weight of the stream of summarize packet from the following host's packet (IP+MAC) in the time frame L0.1. H\_L0.01\_weight is the weight of the stream of summarise packet from the following host's packet (IP) in the time frame L0.01. H\_L5\_weight is the stream of summarize packet from the flowing host's packet (IP) in the time frame L5. The combination that gives the lowest detection accuracy is combination B from naïve bayes and support vector machine. Therefore, feature selections give a certain degree of contribution in order to increase the accuracy of the model.

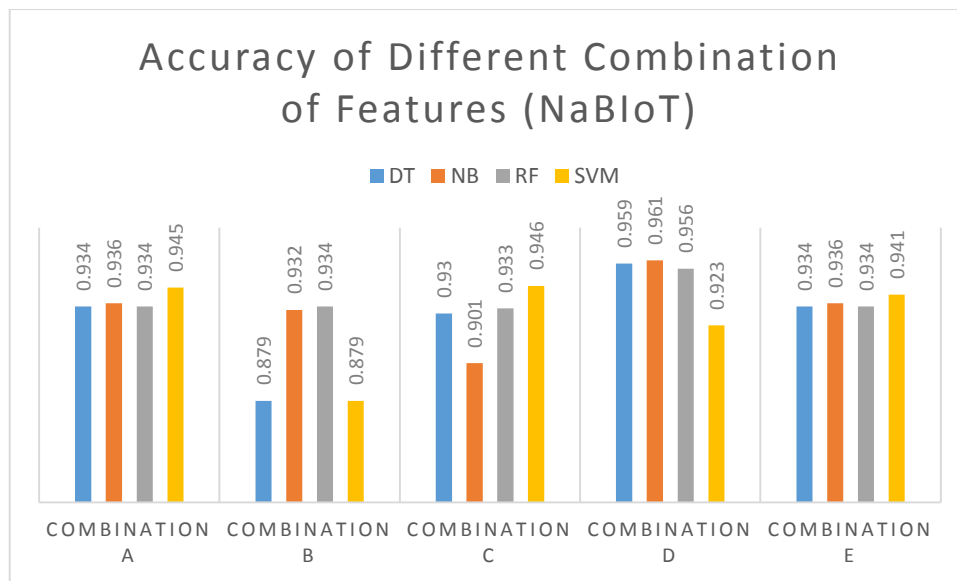


Figure 5(b): Accuracy of Different Combination of Features (NaBIoT)

In addition, the performance matrix of the combinations and classifier for NSL-KDD dataset was measured in false positive rate shown in figure 5(c) and true positive rate shown in figure 5(d). The lowest false positive rate among all of the combinations is combination C from naïve bayes which has a score of 0.00536. The highest true positive rate among all of the combinations is combination A from naïve bayes which has a score of 0.579. Although both of these combinations from naïve bayes give the best result in false positive rate and true positive rate, but they do not give the best result in accuracy, which has been discussed. In terms of overall, combination C from support vector machine has the most balanced score. For combination C from naïve bayes, it does give the best result in false positive rate but the performance for true positive rate is only 0.256 and with the accuracy score of 0.574. For combination A from naïve bayes, it does give the best result in true positive rate but the performance for false positive is only 0.0601. With the comparison in terms of overall in accuracy, false positive rate and true positive rate, the best combinations that perform the best is combination C from support vector machine which has the score of false positive rate of 0.00834, true positive rate of 0.563 and accuracy of 0.748.

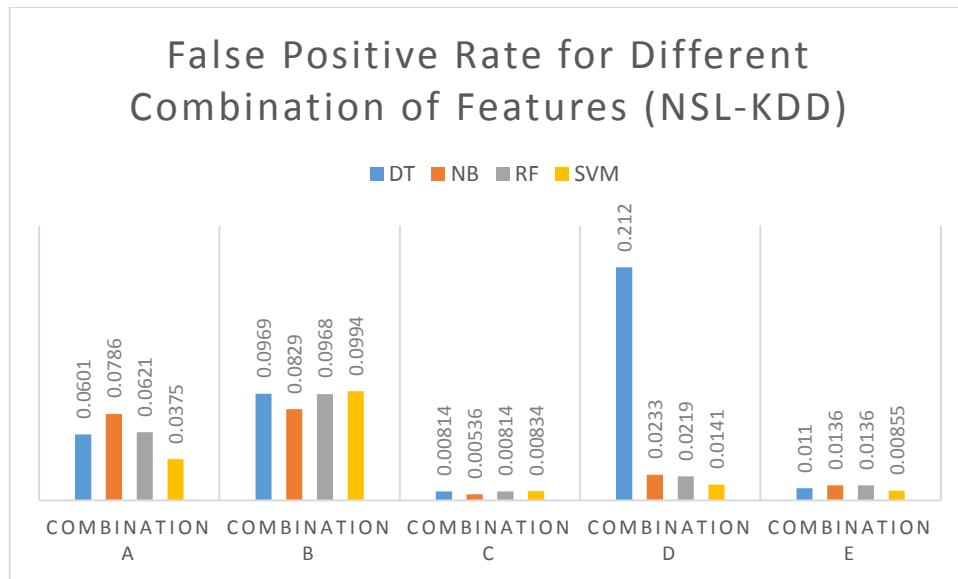


Figure 5(c): False Positive Rate for Different Combination of Features(NSL-KDD)

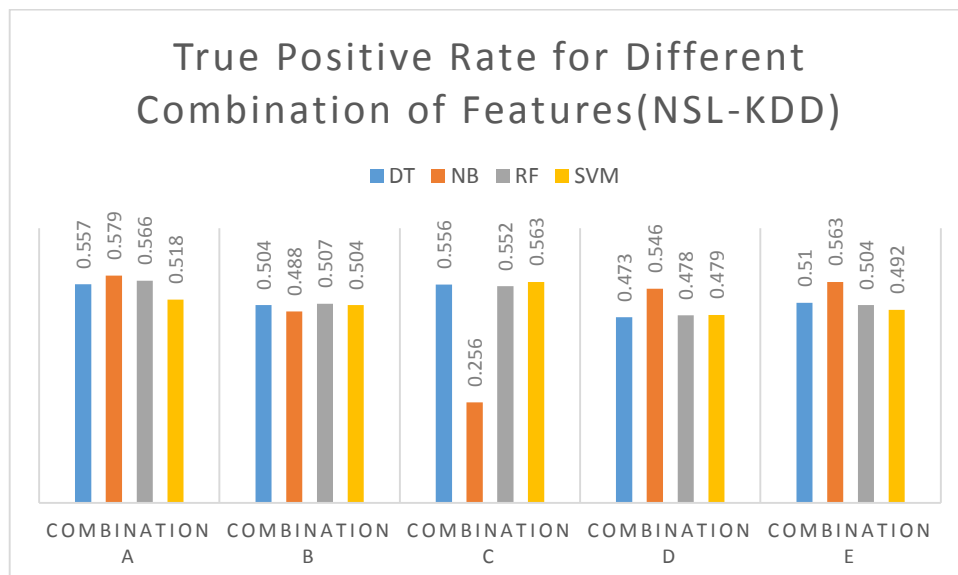


Figure 5(d): True Positive Rate for Different Combination of Features(NSL-KDD)

The performance matrix for combination and classifier for NaBIoT dataset was measured in false positive rate shows in figure 5(e) and true positive rate shows in figure 5 (f). The lowest false positive rate among all of the combination of features is combination C from naïve bayes classifier which has a rate of 0.059. The highest true positive rate among all of the combination of features is combination D from support vector machine classifier which has a rate of 0.967. Although combination C from naïve bayes has the lowest false positive rate but it has a fairly low true positive rate which is

0.897 and accuracy which is 0.901. The true positive rate that has the highest rating is combination D from support vector machine but it also has fairly high false positive rate which is 0.517 and the accuracy is 0.923. In terms of overall, the most well balanced in accuracy, false positive rate and true positive rate is combination D from naïve bayes. It has false positive rate of 0.0738, true positive rate of 0.965 and accuracy of 0.961. Therefore, combination D from naïve bayes can be concluded as the combination and classifier that has the best overall performance.

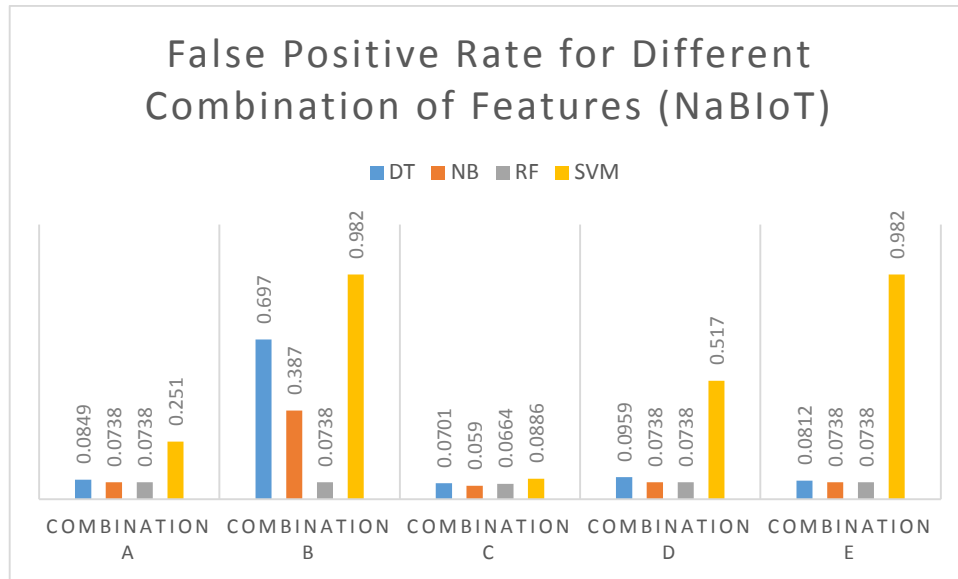


Figure 5(e): False Positive Rate for Different Combination of Features(NaBIoT)

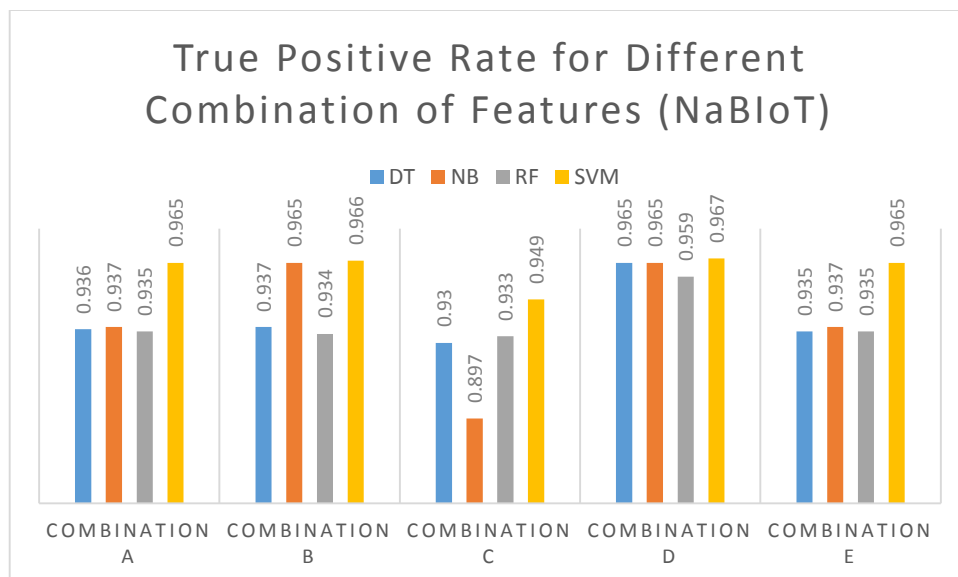


Figure 5(f): True Positive Rate for Different Combination of Features (NaBIoT)

After all of the performance of each combination and classifier was measured, there are one best performance combination and classifier from each dataset. For NSL-KDD, combination C from support vector machine classifier. On the other hand, combination D from naïve bayes classifier perform best in NaBIoT dataset. The false positive rate, true positive rate, accuracy, recall, precision, and f1 score is further investigate in both of this combination and classifier which shows in figure 5(g). Support vector machine classifier in NSL-KDD dataset perform the best in false positive rate, it has a false positive rate of 0.00834 while naïve bayes from NaBIoT has a false positive rate of 0.0738. Naïve bayes from NaBIoT has the highest score in true positive rate which is 0.965 while support vector machine classifier from NSL-KDD has only 0.563. The accuracy of both dataset and classifier is also measured. Naïve bayes from NaBIoT achieve 0.961 in accuracy score while support vector machine classifier from NSL-KDD has only 0.748. The fourth matrix to measure the performance is recall. Naïve bayes from NaBIoT has a better scoring than support vector machine from NSL-KDD. The recall score for naïve bayes from NaBIoT is 0.965 while NSL-KDD only has 0.563. The fifth matrix that used to measure the performance is precision. The precision for naïve bayes from NaBIoT is slightly higher than support vector machine from NSL-KDD which is 0.992 for naïve bayes and 0.989 for support vector. The last matrix that used to measure the performance is f1 score. The f1 score in naïve bayes from NaBIoT is performing better than support vector machine from NSL-KDD. The score for naïve bayes is 0.978 and the score for support vector machine is 0.643. The results from the comparison between NSL-KDD and NaBIoT is NaBIoT perform better than NSL-KDD dataset with the performance matrix measured.

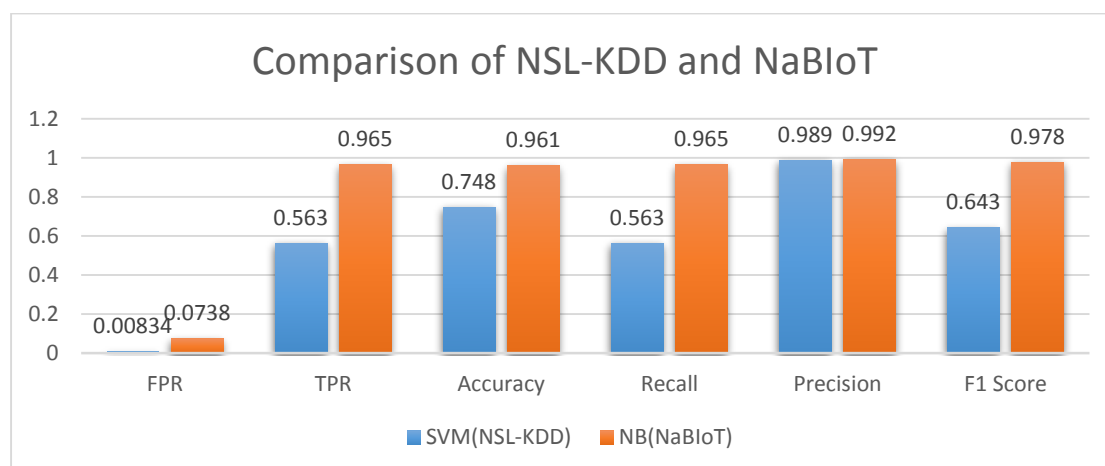


Figure 5(g): Comparison of NSL-KDD and NaBIoT

## Chapter 6: Conclusion

After conducting several test based on different type of classification technique against the test size. The most accurate machine learning classification technique that can be conclude in this research is DT which has the highest mean of 99.9961. the least accurate is RF which have the mean of 99.9635. After conducting it, we are able to have a review on the types of IDS acts differently in different type network. the advantage and disadvantage of different type of IDS has been listed. The proposed solution for this research is by using a machine learning to detect malicious traffic. The data set used to train the model is NSL-KDD and it is further process in the attack type. This enable the model to be train easily when it is in numeric form. The motivation of this project is to have a IDS that are capable to detect most of the attack and decrease the rate of fake alerts. The dataset that perform the best in detecting normal traffic and attack traffic is NaBIoT. The most well balanced in accuracy, false positive rate and true positive rate is combination D from naïve bayes. It has false positive rate of 0.0738, true positive rate of 0.965 and accuracy of 0.961. Therefore, the conclusion that can be made is combination D from naïve bayes can determine as the combination and classifier that has the best overall performance.

## **Bibliography**

- Akash Garg, Prachi Maheshwari, 2016, 'A Hybrid Intrusion Detection System: A Review'
- Aumreesh Ku. Saxena, Dr. Sitesh Sinha, Dr. Piyush Shukla, 2017, 'General Study of Intrusion Detection system and Survey of Agent Based Intrusion Detection System', pp. 417-419
- Anna L. Buczak, Erhan Guven, 2016, 'A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection', vol. 18, issue. 2, pp.1153
- A.T Taha et. al. 2015, 'Behavioral approach for intrusion detection'
- Anuradha & Dr.Gaurav Gupta 2014, 'A Self Explanatory Review Of Decision Tree Classifiers'
- B.I. Santoso, 2016, 'Designing Network Intrusion and Detection System using Signature-Based Method for Protecting OpenStack Private Cloud'
- Ekarat Rattagan, 2016, 'Wi-Fi Usage Monitoring and Power Management Policy for Smartphone Background Applications', pp. 171
- F. Harahap et. al. 2018, 'Implementation of Naïve Bayes Classification Method for Predicting Purchase'
- Gideon Creech, Jiankun Hu, 2014, 'A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguous and Discontiguous System Call Patterns', vol. 63, issue. 4, pp. 807-808
- Geetha V., Dr. Sridhar Aithal, Dr. K. ChandraSekaran, 2016, 'Effect of Mobility over Performance of the Ad hoc Networks' pp. 138-139
- Hang Yang et. al. 2014, 'A Review: The Effects of Imperfect Data On Incremental Decision Tree'
- Ion C. Freeman et al., 2018, 'What are they researching? Examining Industry-based Doctoral Dissertation Research through the Lens of Machine Learning', pp. 1338

- J. Amudhavel et al., 2016, 'A Survey on Intrusion Detection System: State of the Art Review' vol. 9, issue. 11, pp. 4
- Liu Hua Yeo, XiangDong Che, Shalini Lakkaraju, 2017, 'Understanding modern intrusion detection systems: A Survey' pp. 1-4
- L.Dhanabal, Dr. S.P. Shantharajah, 2015, 'A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification'
- Muhammad K. Asif et al., 2013, 'Network Intrusion Detection and its Strategic Importance', pp. 140-141
- M. Bader-El-Den & E. Teitei & T. Perry 2017, 'Biased Random Forest For Dealing With the Class Imbalance Problem' vol. 30 no.7
- M. Peña et al., "ANOVA and Cluster Distance Based Contributions for Feature Empirical Analysis to Fault Diagnosis in Rotating Machinery," 2017 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC), Shanghai, 2017, pp. 69-74.
- N.T Van & T.N Think & L.T. Sach 2017, 'An anomaly-based Network Intrusion Detection System using Deep learning'
- Pratik Satam, 2017, 'Anomaly Based Wi-Fi Intrusion Detection System', pp. 377
- Parveen Sadotra, DR. Chandrakant Sharma, 2016, 'A Review on Integrated Intrusion Detection System in Cyber Security' pp. 24-24
- Radja, 2015 'The overview of wired and wireless networks and the need for the transition from wired to wireless networks' vol. 3, Issue-8, pp.52
- Robert Mitchell, Ing-Ray Chen, 2014, 'A Survey of intrusion detection in wireless network applications', pp.1-9
- S. S.-S. a. S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, New York, NY: Cambridge University Press, 2014, pp. 5-19.
- S.N Sheela Evangelin Prasad et al., 2015, 'Intrusion Detection Systems, Tools and Techniques-An Overview', vol. 8, issue. 35, pp. 1-2



## Bibliography

Sandeep B. Banjale, P.B. Mane, Sandip V. Patil, 2015, 'Wireless LAN Intrusion Detection and Prevention System for Malicious Access Point' pp. 487

Syed Shariyar Murtaza et al., 2013, 'A Host-Based Anomaly Detection Approach by Representing System Calls as States of Kernel Modules', pp. 431- 432

- Syed Shariyar Murtaza et al., 2013, 'A Host-based Anomaly Detection Approach by Representing System Calls as States of Kernel Modules' pp. 431-432
- Shikha Agrawal, Jitendra Agrawal, 2015, 'Survey on Anomaly Detection using Data Mining Techniques', pp. 709
- Sharmilla. S & Shanthi. T 2016, 'A survey on wireless ad hoc network'
- Vandana P et. al. 2014, 'B-DIDS: Mining anomalies in Big-Distributed intrusion detection system'
- W.B. Zulfikar, Y.A. Gerhana, A.F. Rahmania, 2018, 'An Approach to Classify Eligibility Blood Donors Using Decision Tree and Naive Bayes Classifier'
- Xuyang Jing, Zheng Yan, Witold Pedrycz, 2019, 'Security Data Collection and Data Analytics in the Internet: A Survey', pp. 586
- Y. Wang, 2018, 'A Novel Consistent Random Forest Framework: Bernoulli Random Forests'
- Y. EL MOURABIT et al, A Mobile Agent Approach for IDS in Mobile Ad Hoc Network IJCSI International Journal of Computer Science Issues, Vol. II, Issue I, No I, January 2014, pp 148-152.
- Yousef EL Mourabit et. al. 2014, 'Intrusion Detection System In wireless Sensor network Based On Mobile Agent'
- Y. Mehmood et. al. 2015, 'Distributed Intrusion Detection System using Mobile Agents in Cloud Computing Environment
- Yu Liu et. al. 2019, 'An Improved Random Forest Algorithm Based on Attribute Compatibility'
- Zeng-Quan Wang et al.,2016, 'Research on Distributed Intrusion Detection System', pp. 183
- Zhang Changsen, Mao Yan, 2015, 'Study on mine communication network based on Ethernet and WSN', pp.183

- Y. Meidan et al., "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," in *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12-22, Jul.-Sep. 2018.
- Z. A. Bakar, D. I. Ispawi, N. F. Ibrahim and N. M. Tahir, "Classification of Parkinson's disease based on Multilayer Perceptrons (MLPs) Neural Network and ANOVA as a feature extraction," 2012 IEEE 8th International Colloquium on Signal Processing and its Applications, Melaka, 2012, pp. 63-67.

## Appendix

	WLANs	WPANs	WSNs	Ad hoc networks	Mobile telephony	WMNs	CPSs
Signature based	variable CONOP, easy updates	variable CONOP, constrained resources	minimal processing burden	high detection rate	variable CONOP, constrained resources, easy updates	high detection rate	minimal processing burden
Anomaly based	unknown attacks	unknown attacks	minimal persistent storage	unknown attacks	well defined CONOP	well defined CONOP	well defined CONOP
Specification based	unknown attacks	unknown attacks	well defined operation	unknown	unknown attacks	unknown attacks	well defined operation
Reputation based	find selfish actors	find selfish actors	find selfish actors	find selfish actors	find selfish actors	find selfish actors	find selfish actors
Behavior based	low false negatives	minimal memory	low false negatives	minimal memory	minimal memory	minimal memory	low false negatives
Traffic based	metadata rich	metadata rich	less data set error	metadata rich	metadata rich	less data set error	less data set error
Multitrust	expanded data set	expanded data set	credible data set	expanded data set	expanded data set	credible data set	credible data set

Table 1: Mitchell & Chen (2014) Describes the advantages of Intrusion Detection System techniques for wireless networks

	WLANs	WPANs	WSNs	Ad hoc networks	Mobile telephony	WMNs	CPSs
Signature based	dictionary freshness	dictionary size	dictionary size and freshness	dictionary freshness	dictionary size	dictionary freshness	dictionary freshness
Anomaly based	variable CONOP	variable CONOP	high false positive	variable CONOP	revenue impact	high false positive	reliability impact
Specification based	lack common use cases	lack common use cases	costly expert analysis	lack common use cases	lack common use cases	lack common use cases	costly expert analysis
Reputation based	selfish actor sanctions	selfish actor sanctions	selfish actor sanctions	selfish actor sanctions	revenue impact	selfish actor sanctions	reliability impact
Behavior based	erratic profiles	erratic profiles	dormant attacker	erratic profiles	erratic profiles	erratic profiles	dormant attacker
Traffic based	inconsistent visibility	limited storage	limited storage	inconsistent visibility	limited storage	inconsistent visibility	inconsistent visibility
Multitrust	dynamic population	dynamic population	increased storage burden	dynamic population	dynamic population	increased storage burden	federated population

Table 2: Mitchell & Chen (2014) Describes the disadvantages of Intrusion Detection System techniques for wireless networks

# Poster

## A Comprehensive Analysis of Intrusion Detection System in Internet of Things

Teh Boon Seong, Faculty of Information Technology (HONS) Communication and Networking, University Tunku Abdul Rahman

### Introduction

Intrusion detection system (IDS) works as an alarm mechanism for computer system. It detects any malicious activity happened to the computer system and it alerts an alarm message to notify user there are malicious activity. There are IDS that are able to take action when malicious or anomalous network was detected, which include suspend the traffic sent from suspicious IP address.

#### 1.1 Problem Statement

- what is the advantage or disadvantage of different type of IDS, placement strategy, data collection method and the types of network that the IDS should be place

#### 1.2 Project Scope

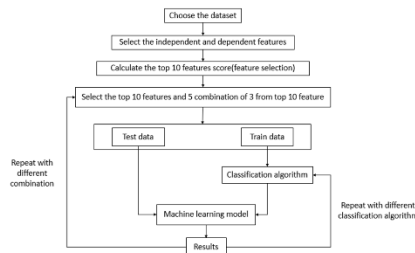
-Review on the type of intrusion detection system, placement strategy, and how the data are collected.  
 -Classification technique will be tested with different test size.  
 -Classification technique will be tested with different dataset.

#### 1.3 Project Objective

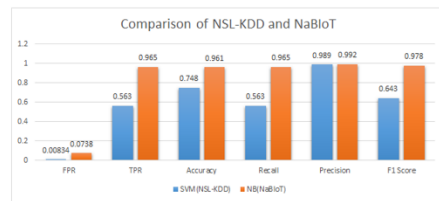
-The main objective for this research is to propose a machine learning IDS which are capable to predict or detect any malicious traffic.  
 -A review on different type of intrusion detection system and their advantage and disadvantage.  
 - To find out the dataset that has great performance in detecting anomaly and normal traffic.  
 - To find out the best feature that have the best performance

### Methods and Materials

To train a model, the data from NSL-KDD is further process. Then 4 machine learning classification technique is used to train the model. The model is trained with NSL-KDD dataset and NaBloT dataset. NaBloT is a wireless dataset. The flow is repeated with different combinations of feature, different classifier and different dataset.



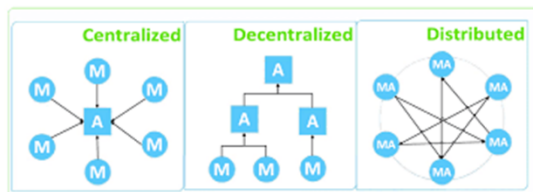
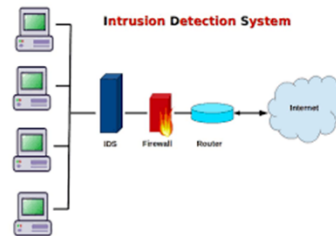
### Discussion



This is the result of comparison between the best performance in combination of features and the type of datasets. As shown in the graph above, naive bayes from NaBloT dataset is performing better than NSL-KDD dataset. Besides, ANOVA feature selection method is selected as the method to calculate the feature score. This method is suitable in this scenario because the input of the dataset is numerical value and the output is categorical value. Numerical value include floating point, which same as the input of the dataset. The output of the dataset is categorical value is Boolean which is either 1 or 0.

### Conclusions

Based on the result from discussion. We are able to conclude that decision tree has the highest accuracy and it is able to detect most of the malicious traffic. Besides, a comprehensive analysis based on the type of intrusion detection system is done in literature review. The classifier, dataset and combination which has the best performance is achieved.



# Weekly Report

## FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3, Year 3	Study week no.: 2
Student Name & ID: Teh Boon Seong 16ACB04462	
Supervisor: DR Vasaki a/p Ponnusamy	
Project Title: A Comprehensive Analysis of Intrusion Detection System in Internet Of Things	

### 1. WORK DONE

- Redefined objectives and project scopes, Chapter 1 and Chapter 2 completed.
- 

### 2. WORK TO BE DONE

- Study a new machine learning classifier
- Find wireless dataset to train the machine learning model

### 3. PROBLEMS ENCOUNTERED

- Unable to find the appropriate wireless dataset to train the model

### 4. SELF EVALUATION OF THE PROGRESS

- Need to find more information about wireless dataset



Supervisor's signature




Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)


<b>Trimester, Year: Trimester 3, Year 3</b>	<b>Study week no.: 4</b>
<b>Student Name &amp; ID: Teh Boon Seong 16ACB04462</b>	
<b>Supervisor: DR Vasaki a/p Ponnusamy</b>	
<b>Project Title: A Comprehensive Analysis of Intrusion Detection System in Internet Of Things</b>	

<b>1. WORK DONE</b> <ul style="list-style-type: none"><li>• Successfully found a wireless dataset which is NaBIoT</li></ul>
<b>2. WORK TO BE DONE</b> <ul style="list-style-type: none"><li>• Complete chapter 1 and 2.</li><li>• Enhance the python code for machine learning in Spyder</li><li>• Find the suitable feature selection method</li></ul>
<b>3. PROBLEMS ENCOUNTERED</b> <ul style="list-style-type: none"><li>• There is some issue regarding the selection of independent feature and dependent feature</li></ul>
<b>4. SELF EVALUATION OF THE PROGRESS</b> <ul style="list-style-type: none"><li>• Need to figure out the solution for feature selection</li></ul>



---

Supervisor's signature



---

Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

<b>Trimester, Year: Trimester 3, Year 3</b>	<b>Study week no.: 6</b>
<b>Student Name &amp; ID: Teh Boon Seong 16ACB04462</b>	
<b>Supervisor: DR Vasaki a/p Ponnusamy</b>	
<b>Project Title: A Comprehensive Analysis of Intrusion Detection System in Internet Of Things</b>	

<b>1. WORK DONE</b> <ul style="list-style-type: none"><li>• Found the suitable feature selection</li><li>• Implemented into the dataset to select the feature</li><li>• Separate some function into independent self-define library</li><li>• Chapter 1 and 2 is done</li></ul>
<b>2. WORK TO BE DONE</b> <ul style="list-style-type: none"><li>• Further study to feature selection.</li><li>• Enhance the machine learning model code.</li></ul>
<b>3. PROBLEMS ENCOUNTERED</b> <ul style="list-style-type: none"><li>• There is some minor error happened when training the machine learning model.</li></ul>
<b>4. SELF EVALUATION OF THE PROGRESS</b> <ul style="list-style-type: none"><li>• Need to study more about feature selection</li></ul>



---

Supervisor's signature



---

Student's signature



# FINAL YEAR PROJECT WEEKLY REPORT

(Project I Project II)

<b>Trimester, Year: Trimester 3, Year 3</b>	<b>Study week no.: 8</b>
<b>Student Name &amp; ID: Teh Boon Seong 16ACB04462</b>	
<b>Supervisor: DR Vasaki a/p Ponnusamy</b>	
<b>Project Title: A Comprehensive Analysis of Intrusion Detection System in Internet Of Things</b>	

<b>1. WORK DONE</b> <ul style="list-style-type: none"><li>• Chapter 3 is done</li><li>• The code for machine learning is done</li></ul>
<b>2. WORK TO BE DONE</b> <ul style="list-style-type: none"><li>• Complete Chapter 5.</li><li>• Fine tune the wireless dataset to have better performance</li><li>• Study about the theory behind the feature selection method selected in this research</li><li>•</li></ul>
<b>3. PROBLEMS ENCOUNTERED</b> <ul style="list-style-type: none"><li>• The machine learning model for wireless dataset is unable to perform well</li></ul>
<b>4. SELF EVALUATION OF THE PROGRESS</b> <ul style="list-style-type: none"><li>• Study what is the errors or causes that make the machine learning model unable to perform well</li></ul>



---

Supervisor's signature



---

Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

<b>Trimester, Year: Trimester 3, Year 3</b>	<b>Study week no.: 10</b>
<b>Student Name &amp; ID: Teh Boon Seong 16ACB04462</b>	
<b>Supervisor: DR Vasaki a/p Ponnusamy</b>	
<b>Project Title: A Comprehensive Analysis of Intrusion Detection System in Internet Of Things</b>	

## 1. WORK DONE

- The error in machine learning model in wireless dataset is fixed.
- Chapter 4 is done

## 2. WORK TO BE DONE

- Complete Chapter 5 and chapter 6.
- Compare the performance machine learning model that is trained with different dataset in more detail

## 3. PROBLEMS ENCOUNTERED

- Unable to find the resource for the feature selection method to include in this research

## 4. SELF EVALUATION OF THE PROGRESS

- Need to learn more detail for feature selection



Supervisor's signature



Student's signature

## Plagiarism



The image shows a screenshot of a plagiarism report interface. At the top, there is a red header with the text "Match Overview" and a close button (X). Below the header, the total match percentage is displayed in large red font as "11%". Underneath, there are navigation arrows (left and right). The main content is a list of 7 sources, each with a colored number, the source name, the source type, and the match percentage. The sources are:

Rank	Source Name	Source Type	Match Percentage
1	pt.scribd.com	Internet Source	1%
2	www.spec2000.net	Internet Source	1%
3	lup.lub.lu.se	Internet Source	1%
4	Robert Mitchell, Ing-Ra...	Publication	1%
5	hdl.handle.net	Internet Source	<1%
6	Shikha Agrawal, Jitendr...	Publication	<1%
7	towardsdatascience.co...	Internet Source	<1%

# Plagiarism

## Turnitin Originality Report

[Document Viewer](#)

Processed on: 23-Apr-2020 13:13 +08  
ID: 1305284537  
Word Count: 16661  
Submitted: 2

fyp By Boon Seong Teh

Similarity Index		Similarity by Source	
<b>11%</b>		Internet Sources:	7%
		Publications:	9%
		Student Papers:	N/A

---

[include quoted](#) [include bibliography](#) [exclude small matches](#) mode: quickview (classic) report Change mode [print](#) [download](#)

1% match (Internet from 19-Feb-2020) <a href="https://pt.scribd.com/document/373769941/scikit-learn-docs-pdf">https://pt.scribd.com/document/373769941/scikit-learn-docs-pdf</a>
1% match () <a href="http://www.spec2000.net">http://www.spec2000.net</a>
1% match (Internet from 13-Dec-2012) <a href="http://lup.lub.lu.se">http://lup.lub.lu.se</a>
1% match (publications) <a href="#">Robert Mitchell, Ing-Ray Chen, "A survey of intrusion detection in wireless network applications", Computer Communications, 2014</a>
<1% match () <a href="http://hdl.handle.net">http://hdl.handle.net</a>
<1% match (publications) <a href="#">Shikha Agrawal, Jitendra Agrawal, "Survey on Anomaly Detection using Data Mining Techniques", Procedia Computer Science, 2015</a>
<1% match (publications) <a href="#">Priyanka Das, Asit Kumar Das, "Rough set based incremental crime report labelling in dynamic environment", Applied Soft Computing, 2019</a>
<1% match (Internet from 10-Apr-2020) <a href="https://towardsdatascience.com/moneyball-linear-regression-76034259af5e?gi=653658abe3c2">https://towardsdatascience.com/moneyball-linear-regression-76034259af5e?gi=653658abe3c2</a>

<b>Universiti Tunku Abdul Rahman</b>			
<b>Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)</b>			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date:	Page No.: 1 of 1



**FACULTY OF INFORMATION AND  
COMMUNICATION TECHNOLOGY**

<b>Full Name(s) of Candidate(s)</b>	TEH BOON SEONG
<b>ID Number(s)</b>	16ACB04462
<b>Programme / Course</b>	CN
<b>Title of Final Year Project</b>	An investigation on internet of things network traffic dataset for intrusion detection

<b>Similarity</b>	<b>Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)</b>
<b>Overall similarity index: <u>11%</u></b>  <b>Similarity by source</b> Internet Sources: <u>7%</u> Publications: <u>9%</u> Student Papers: N/A%	checked and verified
<b>Number of individual sources listed of more than 3% similarity: <u>1%</u></b>	verified
<b>Parameters of originality required and limits approved by UTAR are as Follows:</b> (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are</i>	

Note: Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

***Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.***

  
\_\_\_\_\_

Signature of Supervisor

Name: DR.VASAKI A/P PONNUSAMY

Date: 22 APRIL 2020

\_\_\_\_\_  
\_\_\_\_\_

Signature of Co-Supervisor

Name: \_\_\_\_\_

Date: \_\_\_\_\_



## UNIVERSITI TUNKU ABDUL RAHMAN

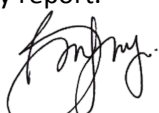

### FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

#### CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	16ACB04462
Student Name	TEH BOON SEONG
Supervisor Name	DR.VASAKI A/P PONNUSAMY

TICK (✓)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
✓	Front Cover
✓	Signed Report Status Declaration Form
✓	Title Page
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
✓	Appendices (if applicable)
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

\*Include this form (checklist) in the thesis (Bind together as the last page)

<p>I, the author, have checked and confirmed all the items listed in the table are included in my report.</p>  <p>(Signature of Student) Date: 23 APRIL 2020</p>	<p>Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.</p>  <p>(Signature of Supervisor) Date: 23 APRIL 2020</p>
---	---

