A BOTNET DETECTION SYSTEM WITH PRODUCT MOMENT CORRELATION COEFFICIENT (PMCC) HEATMAP INTELLIGENT

ONG WEI CHENG

Bachelor of Computer System (Computer System and Networking) with Honours

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : ONG WEI CHENG

Date of Birth

Title : A BOTNET DETECTION SYSTEM WITH PRODUCT MOMENT CORRELATION COEFFICIENT (PMCC) HEATMAP INTTLEGENT

Academic Session : SEMESTER 1 2022/2023

I declare that this thesis is classified as:

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

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

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

Certified by:

____
(Student's Signature)

____
(Supervisor's Signature)

____
New IC/Passport Number
Date: 31 JAN 2023

Ts. Dr. Ahmad Firdaus Zainal Abidin
____
Name of Supervisor
Date: 9 February 2023

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

STUDENT'S DECLARATION

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

(Student's Signature)

Full Name : ONG WEI CHENG

ID Number : CA19098

Date : 31 JAN 2023

A BOTNET DETECTION SYSTEM WITH PRODUCT MOMENT CORRELATION COEFFICIENT (PMCC) HEATMAP INTTLEGENT

ONG WEI CHENG

Thesis submitted in fulfilment of the requirements for the award of the degree of Bachelor of Computer Science in Computer System and Networking

Faculty of Computing

UNIVERSITI MALAYSIA PAHANG

JAN 2023

ACKNOWLEDGEMENTS

First and foremost, I am grateful born into a wonderful family where I have been raised from the beginning of my life. Thank you to my parents and siblings who take care me so much and make me feel so grateful for my childhood and youth.

Next, I would like to express my deepest appreciation to all those who provided me the possibility to complete this project. A special gratitude I give to my supervisor, Dr. Ahmad Firdaus Bin Zainal Abidin for his advice and suggestion, spend time, and guidance on finishing this project report at any time.

Furthermore, not to forget all of my friends that also struggle together in finishing this thesis and doing their best to spend some of their time in order to assist me on this project from the beginning until the end. This project requires a lot of hard work, patience, and time. Moreover, I am very grateful to both of my parents and family for their love and endless support. Therefore, I would like to thank you again for all of them.

Nevertheless, I would like to give my special thanks to the Faculty of Computer, University Malaysia Pahang that gave me an opportunity to infinitely gain knowledge that cannot be found at any other place. I would also like to thank everyone who involved either directly or indirectly in finishing this project by giving their help in any way when developing this project.

ABSTRACT

Botnets must be combated in a concerted manner if they are not to become a danger to global security in the coming years. Botnet detection is currently performed at the host and/or network levels, but these options have important drawback which antivirus, firewalls and anti-spyware are not effective against this threat because they are not able to detect hosts that are compromised via new or malicious software. Therefore, this paper will propose the method and develop a system to detect botnet malware. In order to detect the botnet malware, this study uses feature selection with product-moment correlation coefficient and trains it using decision tree classifier. The botnet detection system is developed according to the decision tree classifier.

ABSTRAK

Botnet mesti diperangi secara bersepadu jika ia tidak menjadi bahaya kepada keselamatan global pada tahun-tahun mendatang. Pengesanan botnet pada masa ini dilakukan di peringkat hos dan/atau rangkaian, tetapi pilihan ini mempunyai kelemahan penting yang mana antivirus, tembok api dan anti-perisian intip tidak berkesan terhadap ancaman ini kerana mereka tidak dapat mengesan hos yang terjejas melalui perisian baharu atau berniat jahat . Oleh itu, kertas kerja ini akan mencadangkan kaedah dan membangunkan sistem untuk mengesan malware botnet. Untuk mengesan perisian hasad botnet, kajian ini menggunakan pemilihan ciri dengan pekali korelasi momen produk dan melatihnya menggunakan pengelas pokok keputusan. Sistem pengesanan botnet dibangunkan mengikut pengelas pokok keputusan.

TABLE OF CONTENT

DECLARATION 3

TITLE PAGE 6

ACKNOWLEDGEMENTS 5

ABSTRACT 6

ABSTRAK 7

TABLE OF CONTENT 8

LIST OF TABLES 11

LIST OF FigureS 12

LIST OF SYMBOLS 14

LIST OF ABBREVIATIONS 15

CHAPTER 1 INTRODUCTION 16

1.1 Introduction 16

1.2 Background of the Problem 16

1.3 Objective 17

1.4 Scope 17

1.5 Thesis Organization 17

CHAPTER 2 LITERATURE REVIEW 18

2.1 Introduction 18

2.2 Three Related Work 20

2.2.1 Symmetrical Uncert Attribute Eval 20

2.2.2 Deep Q-learning based Feature Selection Architecture (DQFSA) 23

2.2.3 Term Frequency Inverse Document Frequency (TF-IDF) 25

2.3 Comparative Analysis 26

2.4 Chapter Summary 28

CHAPTER 3 METHODOLOGY 29

3.1 Introduction 29

3.2 Methodology 29

3.3 Hardware & Software Specification 32

3.4 Dataset 34

3.5 System Development Life Cycle 35

3.6 Functional & Non-Functional Requirements 37

3.7 Constraints & Limitations 37

3.8 Context Diagram 38

3.9 Use Case Diagram & Description 39

3.10 Activity Diagram 40

3.11 Testing Plan 40

CHAPTER 4 RESULTS 41

4.1 Introduction 41

4.2 Implementation 41

4.3 Result 50

4.4 Development 56

4.5 Web Hosting 60

CHAPTER 5 CONCLUSION 76

5.1 Objective Revisited 76

5.2 Limitation 79

5.3 Future Work 79

References **81**

LIST OF TABLES

Table 2.1 Features of SymmetricalUncertAttribute 22

Table 2.2 Specification / Feature of Deep Q-learning based Feature Selection Architecture (DQFSA) 24

Table 2.3 Specification / Feature of Modified Term Frequency Inverse Document Frequency (MTF-IDF). 26

Table 2.4 Advantage & Disadvantage of the existing system 26

Table 2.5 Platform of the existing system 28

Table 2.6 shows the features selection method used by them compared to this study28

Table 3.1 Details of laptops that have been used. 32

Table 3.2 Details of Anaconda Navigator (anaconda3), MATLAB and Visual Studio Code software. 32

Table 3.3 Botnet family 34

Table 3.4 Functional & Non-Functional Requirements 37

Table 3.5 Constraints & Limitation 37

Table 4.1 Accuracy based on the classifier 50

Table 4.2 Accuracy in Decision Tree Classifier 50

Table 4.3 Feature of PMCC 51

Table 4.4 Result and finding. 55

Table Python Library and Description 57

Table 5.1 Features Selection for 1st Approach 76

Table 5.2 Features Selection for 2nd Approach 77

Table 5.3 Features Selection for 3rd Approach 78

Table 5.4 Result of Coarse Tree Classifier 79

LIST OF FIGURES

Figure 2.1 The overview of Feature selection Retrieved from Machine Learning Mastery website 19

Figure 2.2 Workflow of the method 20

Figure 2.3 Result of the performance from different machine algorithm 21

Figure 2.4 Deep Q-learning based Feature Selection 23

Figure 2.5 Assessment measurements results from various 24

Figure 2.6 Android Malware Detection model 25

Figure 2.7 Performance results from various machine 26

Figure 3.1 The methodology of the research and development 30

Figure 3.2 Rapid Application Development diagram 35

Figure 3.3 The context diagram of Botnet Detection System 38

Figure 3.4 The use case diagram of Botnet Detection System. 39

Figure 3.5 The activity diagram for user of Botnet Detection System. 40

Figure 4.1 The code for how to plot the correlation heatmap. 41

Figure 4.2 The uploaded dataset in Jupyter Notebook. 42

Figure 4.3 The python code to generate a heatmap in Jupyter Notebook 42

Figure 4.4 Botnet Heatmap 43

Figure 4.5 Example of Feature Selection method works. 44

Figure 4.6 Best features selection for the first approach 44

Figure 4.7 Best feature selection for the second approach 45

Figure 4.8 Best feature selection for the third approach 45

Figure 4.9 How features are selected 46

Figure 4.10 The features were selected 47

Figure 4.11 Coding of training using classifier of decision tree 47

Figure 4.12 Select the classifier to train 48

Figure 4.13 The accuracy of classifiers 49

Figure 4.14 show the correlative matrix for the first approach 52

Figure 4.15 ROC Curve for the first approach 52

Figure 4.16 show the correlative matrix for the second approach 53

Figure 4.17 ROC Curve for the second approach 53

Figure 4.18 show the correlative matrix for the third approach 54

Figure 4.19 ROC Curve for the third approach 54

Figure 4.20 Confusion matrix table 55
Figure 4.21 Code to build the model 56
Figure 4.22 Code for the Prediction System 56
Figure 4.23 Code for generating a random dataset file. 58
Figure 4.24 Code of user input features value using sidebar. 58
Figure 4.25 Interface of the Botnet Detection System. 59
Figure 4.26 AWS EC2 Dashboard 60
Figure 4.27 List of Instance 60
Figure 4.28 Name the instance 61
Figure 4.29 Create new key pair 62
Figure 4.30 Other Settings 62
Figure 4.31 Security Group 63
Figure 4.32 Create New Security Group 63
Figure 4.33 Update Inbound Rules 64
Figure 4.34 RSA Key Pair 65
Figure 4.35 DNS Public IP 66
Figure 4.36 Command line to access the AWS server. 66
Figure 4.37 EC2 AWS server . 67
Figure 4.38 Install pyhton in the EC2 AWS Server 68
Figure 4.39 Python validation in the EC2 AWS Server 68
Figure 4.40 Git Clone My Repository 69
Figure 4.41 Set Domain Name in Route 53 70
Figure 4.42 Record List 71
Figure 4.43 Public IP Address of Instance 71
Figure 4.44 Assign Public IP Address 72
Figure 4.45 Assign Sub Domain in Record 72
Figure 4.46 GoDaddy My Product Page 73
Figure 4.47 Edit Nameservers 74
Figure 4.48 AWS Naemeserver 74
Figure 4.49 Botnet Detection Website 75

LIST OF SYMBOLS

LIST OF ABBREVIATIONS

DDoS	Distributed Denial of Service
IoT	Internet of Things
PMCC	Product moment correlation coefficient
TPR	True Positive Rate
FRP	False Positive Rate
AI	Artificial Intelligence
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
KNN	K-Nearest Neighbor
SVM	Support Vector Machines

CHAPTER 1
INTRODUCTION

1.1 Introduction

A botnet is a collection of internet-connected devices, such as smartphones, desktop computers, internet of things (IoT) devices, and servers, that have been infected and are controlled by a single malicious programme without the owner's awareness. [1] The term "botnet" is derived from the word "robot" and "network" combined. DDoS attacks, data theft, spam, and giving the attacker access to the device and its connection are all common uses for botnets. Threat actors, mostly cybercriminals, have remote control over infected devices and employ them for certain functions, even if the damaging operations are hidden from the user.

1.2 Background of the Problem

We decided to build a machine learning model to detect the botnet before it spreads. This makes it easier for the network to spot the red flag before it gets worse. The usage of feature selections is critical since there are too many features to choose from, which can lead to overfitting of the model and delayed and inefficient malware detection. As malware threats become more prevalent, so will the threats to user’s personal information. This is extremely concerning and, if not addressed, it is extremely dangerous. As a result, we'll use the botnet dataset from the publication Mobile Botnet Detection: A Deep Learning Approach Using Convolutional Neural Networks in this paper. [2], [3] We decided to use the heatmap and machine learning as the method to detect the Botnet.

1.3 Objective

There are three objectives in this project which are:

1. To study feature selection of Product Moment Correlation Coefficient (PMCC) algorithm with heatmap for machine learning model classification and development.
2. To develop a Botnet detection system with Product Moment Correlation Coefficient (PMCC) with heatmap intelligent.
3. To evaluate the detection performance of the Botnet detection system.

1.4 Scope

To study of the proposed system are listed below:

i) This research is to improve the efficiency and resourcefulness of botnet detection techniques based on machine learning and use the most efficient algorithm to develop the botnet detection system.

1.5 Thesis Organization

Chapter 1 is briefly describing the introduction about malware type which is botnet. Next, it includes problem statements, objectives, scope, and thesis organization.

Chapter 2 will discuss the literature review of the system. This chapter is divided into two sections: existing system research and a comparison of the existing and proposed systems.

Chapter 3 will discuss the methodology used during the development of detecting botnets using heatmap and machine learning. This chapter also covers the hardware, software, and botnet dataset that have been used in this project.

Chapter 4 will discuss the implementation, results, and development. This chapter will also go into the function, how the procedure was done, and the outcome of the suggested system. In addition, the testing results will be supplied. After the testing is down, the system development will start using the result that we get is testing.

Lastly, Chapter 5 is the objective overview, the limitation and discussed for any future enhancement for the methodology and the algorithm.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

Machine learning is an artificial intelligence (AI) technique that allows systems to learn and improve from their experiences without having to be explicitly programmed. Machine learning is concerned with the development of computer programmes that can access and utilise data and learn on their own.

The learning process starts with observations or data, such as examples, direct experience, or instruction, in order to seek for trends in data and make informed decisions in the future. The main objective is to enable computers to learn on their own and adapt their behaviour accordingly without the need for human intervention.

Furthermore, there are just a few reasons why feature selection approaches are used. Shorter training times, improved generalisation by eliminating overfitting, and model simplification to make them easier to read and improve accuracy are just a few examples.

Next, the common algorithms that are being used for feature selection inside machine learning are supervised or unsupervised. Figure 2-1 shows an overview of the feature selection method.

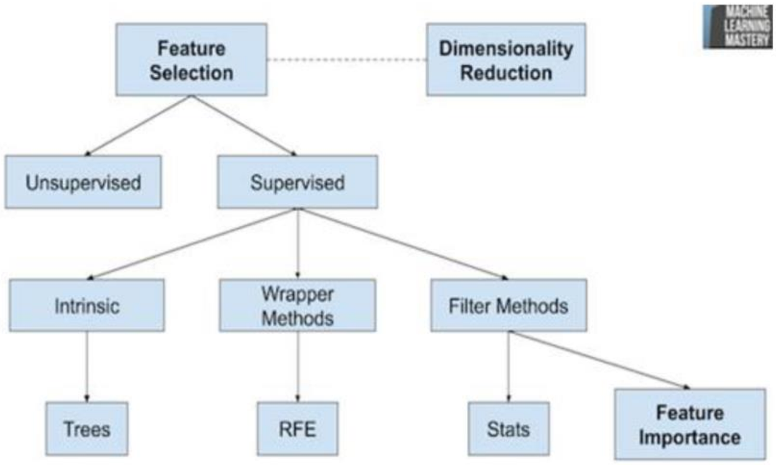


Figure 2.1 The overview of Feature selection Retrieved from Machine Learning Mastery website

By maintaining the previous dataset, supervised methods allow the machine to forecast the feature. When we want the machine to investigate the data in order to generate hypothesis from an unlabelled dataset in order to explain subliminal structures, we utilise an unsupervised algorithm. The combination of supervised and unsupervised algorithms in a semi-supervised algorithm allows the computer to interact with both datasets (either labelled or unlabelled). This algorithm is typically employed when the dataset necessitates the machine being educated, found, or skilled. A reinforcement algorithm is a technique for allowing robots to immerse themselves in their surroundings in order to learn through trial and error or rewards..

We also provide three examples of existing work in this paper. These three existing works have been studied by different groups. These three works have shown different results by using the same features.

2.2 Three Related Work

2.2.1 Symmetrical Uncert Attribute Eval

Symmetrical Uncert Attribute Eval H al-kaaf, A Ali, S Shamsuddin and S Hassan proposed to use 3 types of different feature selection method which is sequential minimal optimization. SMO), Decision Tree (J48) and Naive Bayes which achieved highest accuracy of 0.88 and precision of 0.910 when combining with Symmetrical Uncert Attribute Eval.[4]

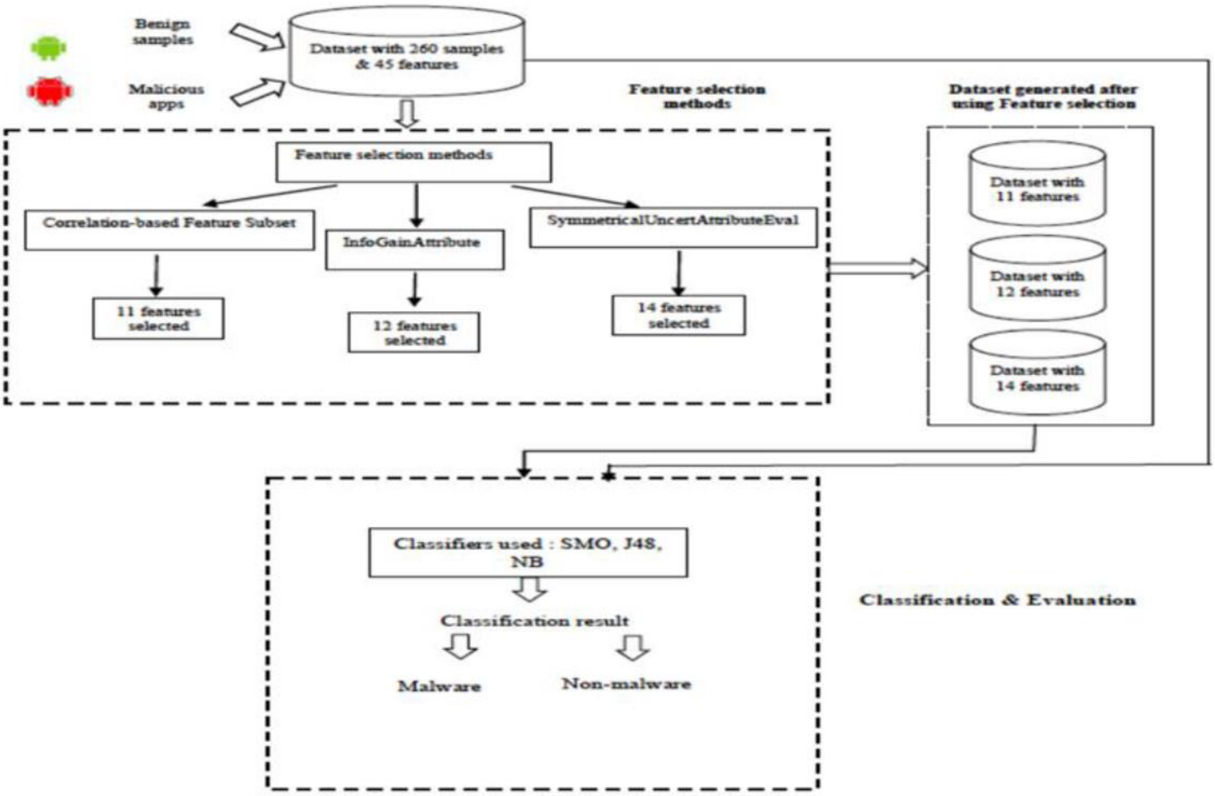


Figure 2.2 Workflow of the method

Figure 2.2 shows how they apply the machine learning the classifier orderly to achieve significant results. This study made use of a malicious dataset collected from PROGuard and Drebin. The permissions for the apps are extracted using static extraction. Correlation-based Feature Subset Selection (CFS), InfoGainAttribute, and SymmetricalUncertAttribute are the feature selection methods used.

Correlation-based Feature Subset Selection (CFS) is a channel calculation approach that evaluates the expectation of each trait in terms of repetition and the relationship between them. It selects highlights that have a strong link with the class. InfoGainAttribute is a type of channel process that evaluates the inclusion based on the estimation of its data pick up concerning the class. SymmetricalUncertAttribute evaluates the highlights based on the balanced vulnerability of each property. The SymmetricalUncertAttributeEval estimation is either zero or one, with one indicating that the trait or highlight is relevant to the class and 0 indicating that the characteristic is irrelevant to the class..

The Weka tool is used for the evaluation component to calculate Overall Accuracy, False Positive Rate, and Precision. One of the measurements used to evaluate grouping models is accuracy. The False Positive Rate (FPR) quantifies the number of negatives that are incorrectly identified as positives (for example the level of clean applications that misclassified as malware applications) Whereas TP refers to the number of malware applications that delegated malware applications, FN refers to the number of clean applications that were incorrectly labelled spiteful. TN refers to the number of thoughtful applications that have been delegated favourably. FN refers to the number of irregular applications that have been mislabeled as ordinary. Precision quantifies the number of negatives that are incorrectly identified as certain (for example the level of clean applications that are misclassified as malware applications).

Classifiers algorithms	Features	ACC	TPR	FPR	PREC
NaiveBayes	45f	0.85	0.869	0.169	0.837
	11f	0.857	0.862	0.146	0.855
	12f	0.865	0.869	**0.138**	0.863
	14F	**0.869**	0.877	**0.138**	0.864
SMO	45f	0.876	0.862	0.108	0.889
	11f	0.876	0.838	**0.085**	0.908
	12f	0.8807	0.846	**0.085**	0.909
	14F	**0.884**	**0.854**	**0.085**	**0.910**
	45f	0.792	0.769	0.185	0.806
J48	11f	0.773	0.792	0.246	0.763
	12f	0.776	0.785	0.231	0.773
	14F	0.780	0.792	0.231	0.774

Figure 2.3 Result of the performance from different machine algorithm

Figure 2.3 depicts the outcomes for all of the feature selection approaches. SymmetricalUncertAttributeEval evaluates the value of a quality by evaluating the class's even vulnerability. Although SymmetricalUncertAttribute with SMO and NaiveBayes classifiers produced good results, J48 has low accuracy and an exaggerated FPR.

They came to the conclusion that SymmetricalUncertAttribute is the best employing SMO, with an accuracy of 88.4615%.

Table 2.1 Features of SymmetricalUncertAttribute

Feature Selection Method	SymmetricalUncertAttribute
Classifiers	NaiveBayes, Sequential Minimal Optimization (SMO), Decision Tree (J48)
Highest accuracy	88.46%

2.2.2 Deep Q-learning based Feature Selection Architecture (DQFSA)

This approach e trains an expert using Q-figuring out how to increase the morpheme's normal precision on an approval dataset by sequentially collaborating with the highlighted region. Based on a –greedy investigation technique and experience replay, the specialist studies a vast yet constrained space of feasible activities and repeatedly finds options with improved execution on the learning task. [5]

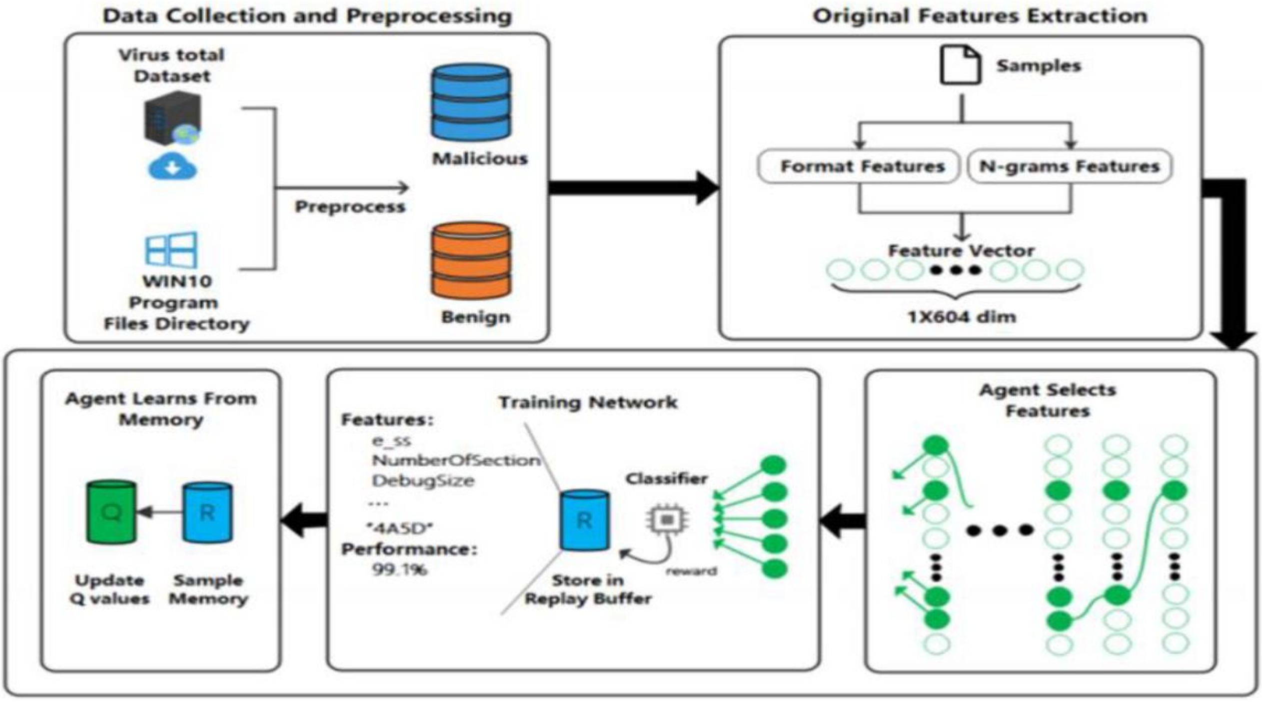


Figure 2.4 Deep Q-learning based Feature Selection

The primary tasks for this model are to develop a learning process. Specialist in selecting highlights sequentially for classification. The assumption that a component performs well in one arrangement mission should be associated with the outcome of another arrangement mission, so that the component option period can be displayed as a Markov Decision Process. Under the -insatiable technique, the specialist selects highlights in a sequential manner until it reaches an end state.

Accuracy / Features Used / Classifier	5	6	7	8	9	10	11	12	13	14	15
KNN	95.21%	97.69%	96.20%	98.80%	99.32%	99.48%	**99.53%**	98.63%	98.70%	99.27%	99.06%
Decision Tree	96.48%	96.76%	96.03%	97.66%	96.60%	98.09%	98.99%	**99.03%**	98.13%	98.23%	98.34%
Random Forest	96.04%	95.96%	98.16%	97.17%	98.59%	98.61%	**99.46%**	98.68%	99.01%	98.87%	99.34%
Naive Bayes	95.90%	94.31%	96.44%	99.25%	98.26%	97.71%	99.17%	99.20%	**99.41%**	98.82%	99.50%
SVM	95.33%	96.86%	**99.29%**	99.15%	97.74%	97.50%	99.20%	98.89%	98.89%	98.89%	98.89%

Figure 2.5 Assessment measurements results from various

They applied 5 classifiers inside the machine to receive the best accuracy. The classifiers that are being used in K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Naive Bayes, Support-Vector Machines (SVM). The detection for detecting malware that they achieved using this method is 99.53%.

Table 2.2 Specification / Feature of Deep Q-learning based Feature Selection Architecture (DQFSA)

Feature Selection Method	Deep Q-learning based Feature Selection Architecture (DQFSA)
Classifiers	K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Naive Bayes, Support-Vector Machines (SVM)
Highest accuracy	99.53%

2.2.3 Term Frequency Inverse Document Frequency (TF-IDF)

Nurul Hidayah Mazlan and and Isredza Rahmi A Hamid implement feature selection algorithm for android malware detection using Term Frequency Inverse Document Frequency (TF-IDF) in Evaluation of Feature Selection Algorithm for Android Malware Detection article. However, as stated in the article, Inverse Document Frequency (IDF) is ignorant during class label training and will produce incorrect weight values in some features. As a result, they proposed a modified version of Frequency Inverse Document Frequency (TF-IDF) to calculate the impact of key malware highlights selected in the Android application testing. Figure 2.6 depicts how the detection model works with the Modified Term Frequency Inverse Document Frequency feature selection (MTF-IDF). [6]

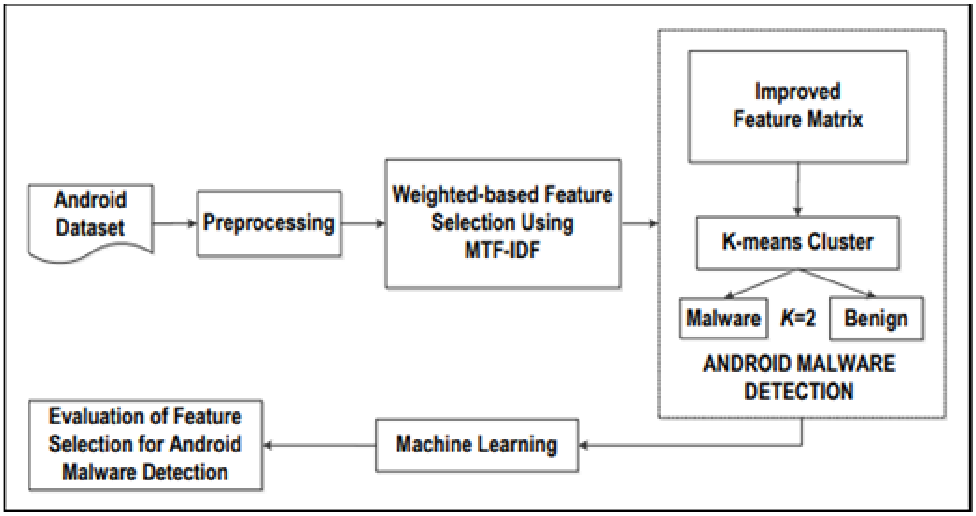


Figure 2.6 Android Malware Detection model

The feature selection process will be used to a dataset of Android information that has been divided into XML document configuration. The information base is dissected to yield the element vector portions. The element determination measure will reduce the unimportant and excessive highlights. At the same time, the highlights used are classified into two types: API call and dangerous consent.

Algorithm	Experiment	TP Rate	FP Rate	Accuracy(%)
Bagging (meta)	TF-IDF	0.954	0.046	95.4
	MTF-IDF	0.976	0.032	97.6
Decision Table (rules)	TF-IDF	0.950	0.047	95.0
	MTF-IDF	0.968	0.040	96.8
Random Forest (tree)	TF-IDF	0.976	0.026	97.6
	MTF-IDF	0.989	0.012	98.9

Figure 2.7 Performance results from various machine

Figure 2.7 depicts the performance gained for Frequency Inverse Document Frequency (TF-IDF) and Modified Frequency Inverse Document Frequency (MTF-IDF) utilising three algorithms: bagging, decision tables, and random forests. In a nutshell, Modified Term Frequency Inverse Document Frequency (MTF-IDF) has the highest accuracy for three algorithms which is 97.6%, 96.8% and 98.9% respectively.

Table 2.3 Specification / Feature of Modified Term Frequency Inverse Document Frequency (MTF-IDF).

Feature Selection Method	Modified Term Frequency Inverse Document Frequency(TF-IDF)
Classifiers	Bagging, Decision Table, Random Forest
Highest accuracy	98.90%

2.3 Comparative Analysis

Here are the advantage and disadvantages of the three related work.

Table 2.4 Advantage & Disadvantage of the existing system

Machine Learning	Advantages	Disadvantages
Symmetrical Uncert Attribute Eval	They applied benign and malware dataset into the machine learning for better results.	Have the lowest accuracy among other researches.

Deep Q-learning based Feature Selection Architecture (DQFSA)	Have the highest accuracy among other researches.	Does not use android malware dataset to train the machine.
Term Frequency Inverse Document Frequency (TF-IDF)	All classifiers have a minimum accuracy of 95.0	Does not train the machine learning using a benign dataset.

Table 2.5 Platform of the existing system

Name	Platform
SolarWinds Security Event Manager	Software
DataDome	Cloud-based
ClickCease	Software

2.4 Chapter Summary

To summarise, this chapter has focused on the creation of three feature selection models by other academics. The literature review demonstrates how they used feature selection to create a powerful detection model. Every consequence and finding of their research is presented, as well as figures and tables.

Table 2.6 shows the features selection method used by them compared to this study

Feature Selection	Paper 1	Paper 2	Paper 3	This Study
Correlation-based Feature Subset Selection (CFS), InfoGainAttribute & SymmetricalUncertAttribute	✓			
Deep Q-learning		✓		
Modified Term Frequency Inverse Document Frequency			✓	
Product Moment Correlation Coefficient (PMCC) + Heatmap				✓

Table 2.6 shows the comparison of features selection method that have been conduct in the existing system of previous researches between this study.

CHAPTER 3
METHODOLOGY

3.1 Introduction

This chapter consists of four stages of methodology: data collection, correlation matrix with heatmap, features selection extraction, and machine learning classifiers. In the data collection process, we use botnet malware and a clean dataset. Then, we conduct a correlation matrix with heatmap. In order to select the best features, we using the method of product moment correlation coefficient (PMCC). Finally, we evaluate the features by using machine learning classifiers, in order to compare the accuracy of the malware detection. Next, this chapter also includes the details of hardware, software and botnet dataset that have been used throughout this research.

The Product Moment Correlation Coefficient (PMCC), which is represented by the symbol r, is a metric for the strength of a linear relationship between two variables. The Pearson correlation coefficient, or r, measures how far away all of these data points are from the line of best fit that a Pearson product-moment correlation attempts to draw across the data of two variables.

The heatmap, which graphically describe data by colouring values, it is simple to see and quickly comprehend complex data. Although modern heatmaps are typically made using specialist heatmapping software, they can also be created manually.

3.2 Methodology

This paper consists of four main phases of methodology: literature review, model development process, evaluating the mode, and system development.

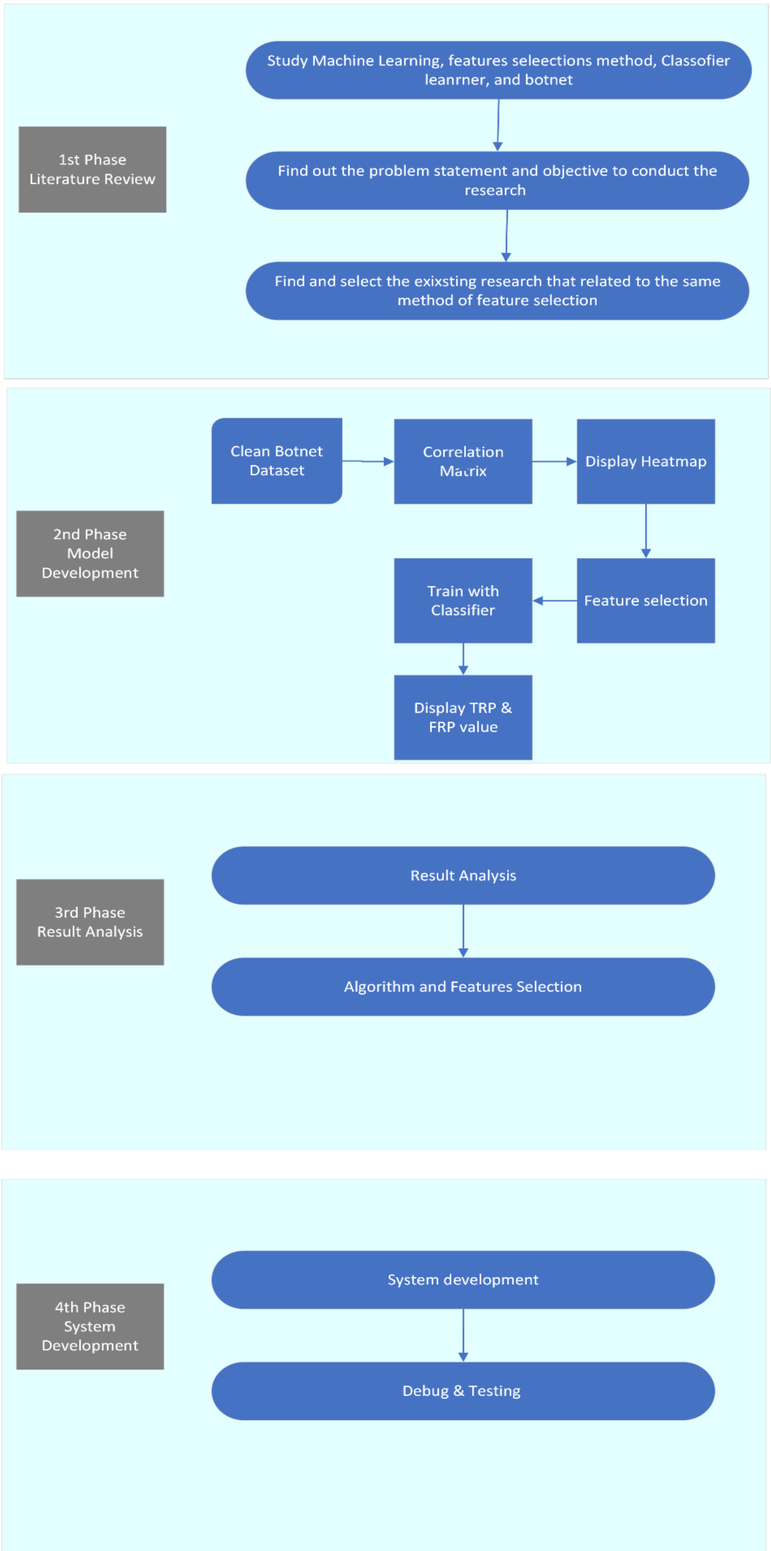


Figure 3.1 The methodology of the research and development

In the 1st phase, which is the literature review, we have started the research by studying more about machine learning, feature selection method that will be chosen, classifier learner that will be used, and understanding the concept of botnet to achieve the best model.

Next, at the 2nd phrase is the model development process. This part consists of four stages which are dataset collection, feature selection, features extraction, and machine learning classifiers. In the data collection process, we chose botnet malware and clean dataset from Android botnet detection dataset for machine learning, figshare. The dataset CSV will be imported into python using Jupyter Notebook. Then, we conduct a correlation matrix to generate a heatmap. Next, in order to select the best features, the product moment correlation coefficient algorithm has been implemented. We will evaluate the dataset to ensure that the best features are selected. Finally, we trained the selected features by using machine learning classifiers, in MATLAB in order to compare the accuracy of the malware detection. We will train the features in three different classifiers such as decision tree, nearest neighbor (KNN), and support vector machines (SVM). The accuracy, TPR, and FRP values were been taken and evaluated.

At 3rd Phase which is the Evaluate Model, which will analyse and make a conclusion for the model's outcome. We have all of the results we require at this point, and we will choose the most effective model to continue with the system development process.

In the 4th Phase, system development will begin using python language to develop the botnet detection system in web applications. Debug and testing will be carried out until it successfully runs and detects botnet viruses.

3.3 Hardware & Software Specification

This subtopic will explain about the hardware and software that are been used for this research in detail. There one hardware and three type of software were used throughout the whole research. Table 6 shows the specification for my laptop and Table 7 show the specification Anaconda Navigator (anaconda3), MATLAB software and Visual Studio Code software that was used to develop the process of data.

Table 3.1 Details of laptops that have been used.

Name	Version	Description	Purpose of use
Laptop (Acer)	Windows 10 64bits	A gaming notebook that can easily bring along and has various functions that can be used in different environment	To write report and thesis, create a heatmap and train the dataset using machine learning classifier. Develop a botnet detection with PMCC

Table 3.2 Details of Anaconda Navigator (anaconda3), MATLAB and Visual Studio Code software.

Name	Version	Description	Purpose of use
Anaconda Navigator (Anaconda3)	Python version 3.10.6	Dissemination of the Python and R programming dialects for logical figuring, that expects to disentangle bundle the board and sending. The dissemination incorporates information science bundles reasonable for Windows, Linux, and macOS	To build a heatmap based on python or R language.
MATLAB	Original License of R2022b version	MATLAB is a programming and numeric computing platform used by millions of engineers and scientists to analyse data, develop algorithms, and create models.	To train the selected features using classifier learner.

Visual Studio Code	Version 1.67.2	Visual Studio Code is a lightweight but powerful source code editor for Windows, macOS, and Linux that runs on your desktop. It includes built-in support for JavaScript, TypeScript, and Node.	To develop the system using python language

3.4 Dataset

Table 3.3 Botnet family

Botnet Family	Number of samples
Anserverbot	244
Bmaster	6
Droiddream	363
Geinimi	264
Misosms	100
Nickyspy	199
Notcompatible	76
Pjapps	244
Pletor	85
Rootsmart	28
Sandroid	44
Tigerbot	96
Wroba	100
Zitmo	80
Total	**1929**

In this study we used the Android dataset from [2], which is known as the ISCX botnet dataset. The ISCX dataset contains 1,929 botnet apps and 4,873 clean apps. The botnet apps were from 14 different families and have been used in previous works including [7][8][9][10][11][12]. The botnet families are shown in Table 8.

3.5 System Development Life Cycle

The system will begin developed by using python language. The system development process will use Anaconda software and Visual Studio Code.

Figure 3.2 Rapid Application Development diagram

From the Figure 3.2 there are 4 phases in RAD which is analysis and quick design, prototype cycles, testing and implementation.

First, analysis and quick design phase is a critical step for the ultimate success of the project, the discussion between supervisor and student is needed to determine the goals and expectations for the project as well as current and potential issues that would need to be addressed during the build. The potential user of the system which is computer users that need to define and finalize their requirements. The computer user needs to upload the file on the website.

Second, in prototype cycles there are 3 step that need to go through, they are demonstrate, refine and build. After quick design is complete, it will demonstrate to supervisor. The information of requirement gathered during the analysis and quick design phase is demonstrate and analysed to define a set of clear data objects crucial for the business. They will give some idea to developers to meet their requirement and developers will refine the design and build it again to meet the requirement. Instead of following the requirements, student will create prototypes with different features and functions and then show them to the supervisor to decide what should and should not have. This process will be repeated until the supervisor is satisfied with the design.

After that testing process is perform for validation requirements. This step requires to test the product and ensure that all part meet the expectations. Feedback needed after testing for any changes or enhancements which is what’s good, what’s not, what works, and what doesn’t is shared. This phase like prototype phase that these two steps are repeated until a final product can be realized that fits both the developers and stakeholder requirements.

The last process is implementation which is the system will take place in the real environment. This phase where the finished product will move to the programming components to a live production environment to conduct comprehensive testing.

3.6 Functional & Non-Functional Requirements

Table 3.4 Functional & Non-Functional Requirements

Functional Requirement	Non-Functional Requirement
1.The Botnet Detection System should enable users to upload the file into the system in order to check their file.	1.The Botnet Detection System should run in web-based application
2.The Botnet Detection System should enable show the result of the uploaded file	2. The Botnet Detection System should run in 24 hours per day and 7 days per week.

3.7 Constraints & Limitations

Table 3.5 Constraints & Limitation

Constraints	Limitations
1.The different algorithm have different featured selection to detect the botnet viruses.	1.The Botnet Detection System model might be dependent on one dataset which may come out with wrong result.
2. Dataset used to train will affect the detection model.	2. The algorithm used may not the most effective algorithm to detect the botnet viruses.

3.8 Context Diagram

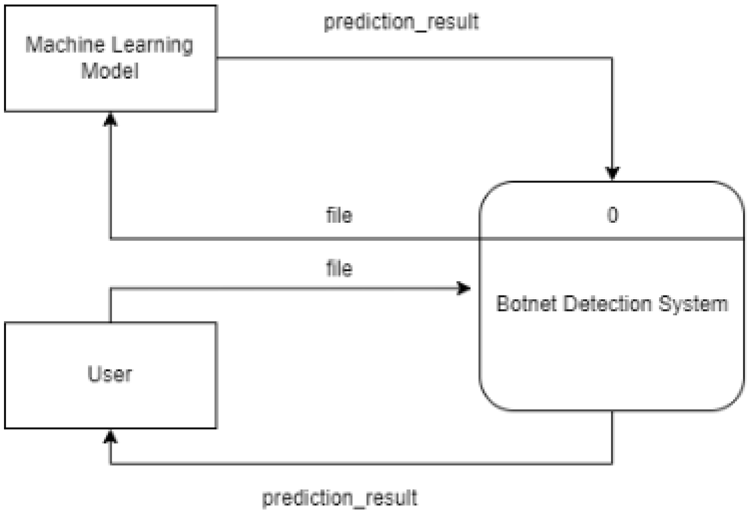


Figure 3.3 The context diagram of Botnet Detection System

In Figure 3.3, user will upload the file to Botnet Detection System. The Botnet Detection System will use the Machine Learning Model to scan the uploaded file by user and get the prediction result. Next the prediction result will send to user.

3.9 Use Case Diagram & Description

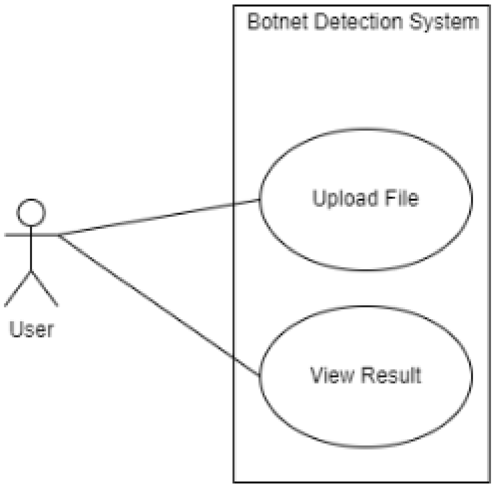


Figure 3.4 The use case diagram of Botnet Detection System.

Use Case ID	Upload File
Brief Description	This use case is to manage user to upload file into the Botnet Detection System. It is indicated user only.
Actor	User
Basic Flow	1. The use case begin when users open the Botnet Detection System. 2. Users are required to upload the file in the interface. 3. The use case end.
Exception Flow	E1:Wrong upload file 1. Users had uploaded wrong file. 2. User reselect the file. 3. The use case return to step number 2 in the basic flow.

3.10 Activity Diagram

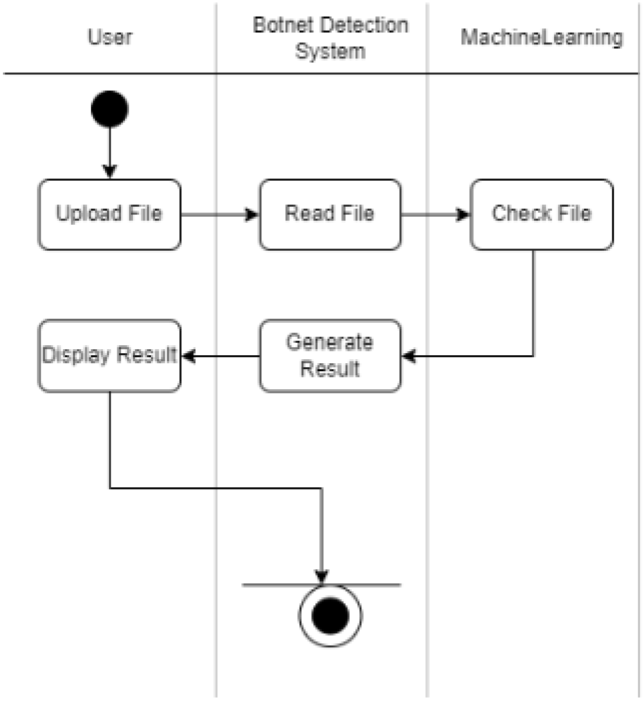


Figure 3.5 The activity diagram for user of Botnet Detection System.

In Figure 3.5, the activity will start at user which they will need to upload the file. Then the Botnet Detection System will read the file and send to Machine Learning to do the checking. Then Botnet Detection system will generate the result for the file and display the result for the user.

3.11 Testing Plan

The testing plan for Botnet Detection System will test the effectiveness of the model that had be trained and the it's accuracy on detecting botnet. Other than that, the testing will also include the functionality of each interface and database.

CHAPTER 4
RESULTS

4.1 Introduction

This chapter will go over the implementation of this research and the development of the system in great detail. All the processes, workspace, and development have been done in two types of software, which are Jupyter Notebook, Anaconda, Matlab, and Visual Studio Code. These tasks will be well explained in the implementation part, while all results will be detailed in the result part.

4.2 Implementation

Implementation process in Jupyter notebook (anaconda3)

```
#import packages
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
data=pd.read_csv("C://Users//bryan//OneDrive - ump.edu.my//FYP//botnets.csv")
```

Figure 4.1 The code for how to plot the correlation heatmap.

We used to import the necessary packages and libraries into Anaconda to make sure the algorithm could run without any problems. Inside the Jupyter Notebook, we will use the pandas, seaborn, and matplotlib packages and libraries. Next, the dataset will be added to the Jupyter notebook by importing the CSV file into the environment using pandas.

```
data
```

	PRO	LOC	F.LOC	COM	M.LOC	STATE	SURF	WIFI	A.MNG	VOICE	...	\/bin	insmod	stdout	stderr	killall	reboot	\/net	\/sys	pminstall	Result
0	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
2	0	1	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	1	0	1	0	0	...	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6797	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6798	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6799	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6800	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6801	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

6802 rows × 343 columns

Figure 4.2 The uploaded dataset in Jupyter Notebook.

To ensure that the correct dataset is imported, a preview of the dataset will be performed. The number of rows and columns is displayed in the dataset preview at the bottom. 343 columns and 6802 rows are present.

```python
#get correlations of each features in dataset
corr = data.corr()
plt.figure(figsize=(300,300))
```

Python

```python
#plot heatmap
corr.style.background_gradient(cmap='coolwarm').set_precision(2)
```

Python

Figure 4.3 The python code to generate a heatmap in Jupyter Notebook

Once the imported data is accurate, a correlation matrix is created to generate a heatmap. The heatmap's figure size and background have been configured. Next, we set the precision of the dataset values in the heatmap to 2 decimal places. Following the computation of the correlation matrix, a heatmap will be generated.

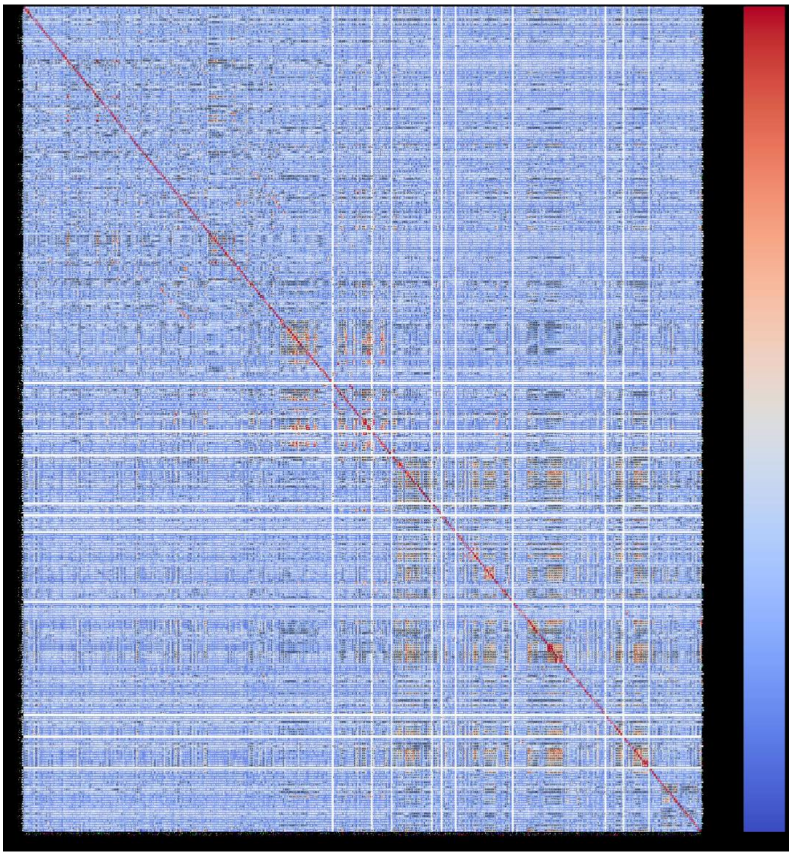


Figure 4.4 Botnet Heatmap

Figure 4.4 shows the whole visual of the botnet heatmap. In furthermore, we use the AJ Pearson correlation technique, often known as the product-moment correlation coefficient (PMCC), to determine the correlation between the characteristics. Then, we will select the best characteristics that are highly associated with one another for usage as model properties.

Feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model creation. It is also known as variable selection, attribute selection, and variable subset selection. We considered using feature selection as one of our primary ideas in machine learning in order to train the machine more efficiently. Figure 4.5 shows an example on how feature selection works in order to achieve the best accuracy and result.

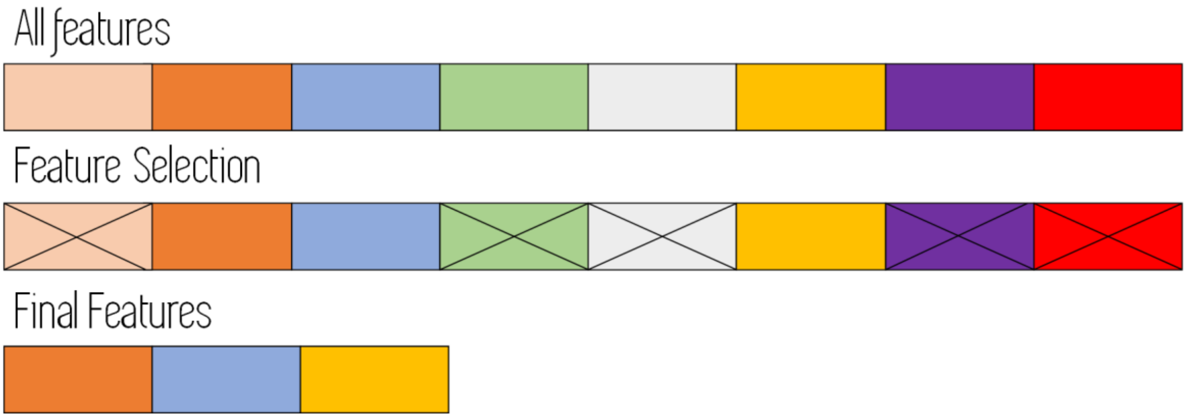


Figure 4.5 Example of Feature Selection method works.

Feature Selection is known as a cycle where you consequently or physically select these highlights which contribute most to your forecast variable or yield in which you are keen on. Having irrelevant or unused data in a machine in order to train them to track better for a specific reason can cause a decline in inaccuracy which results in poor and unreliable results. By distinguishing the irrelevant dataset, the machine can work more effectively and reach the main goal accurately [14],[15].

We have implemented feature selection for the multiple ways of in using the product-moment correlation coefficient. Figures 4.6 to 4.8 illustrate how the best features were chosen. We started the first round of feature selection with the first approach.

Figure 4.6 Best features selection for the first approach

Figure 4.6 shows the number of best features in a Botnet heatmap. For the first approach, we had decided to select the best features based on the value of 0.99, which is the highest value. There were 14 features were successfully been detected. All the best features information will be gathered.

We proceed with another implementation by using the same step and method of feature selections for the second approach. The result will be shown in the next figure.

Figure 4.7 Best feature selection for the second approach

Figure 4.7 also showed the best features in the heatmap for the second approach. The best value that we decide for this approach is 0.90 until 0.99. It is because the more features the more accuracy will be get so that the efficient model will be produced. There were 36 features altogether. All the best features information will be collected for another step purposed.

We proceed with the last implementation by using the same step and method of feature selection for the third approach. The result will be shown in the next figure.

Figure 4.8 Best feature selection for the third approach

Figure 4.8 shows the best features in the heatmap for the last approach which is the third. We decide to add more features to this approach compared to the two previous approaches by choosing the best value starting from the lowest of 0.85 until 0.99. This is one of the steps to get the best accuracy result by training the feature selection. The 47 features were successfully detected.

Implementation process in MATLAB

Once we have compiled the best features from the various approaches, we will proceed to the next step. The next step in MATLAB involves importing the dataset in order to select the previously selected features. We selected the best features manually, one by one, based on the references of the best collections of features.

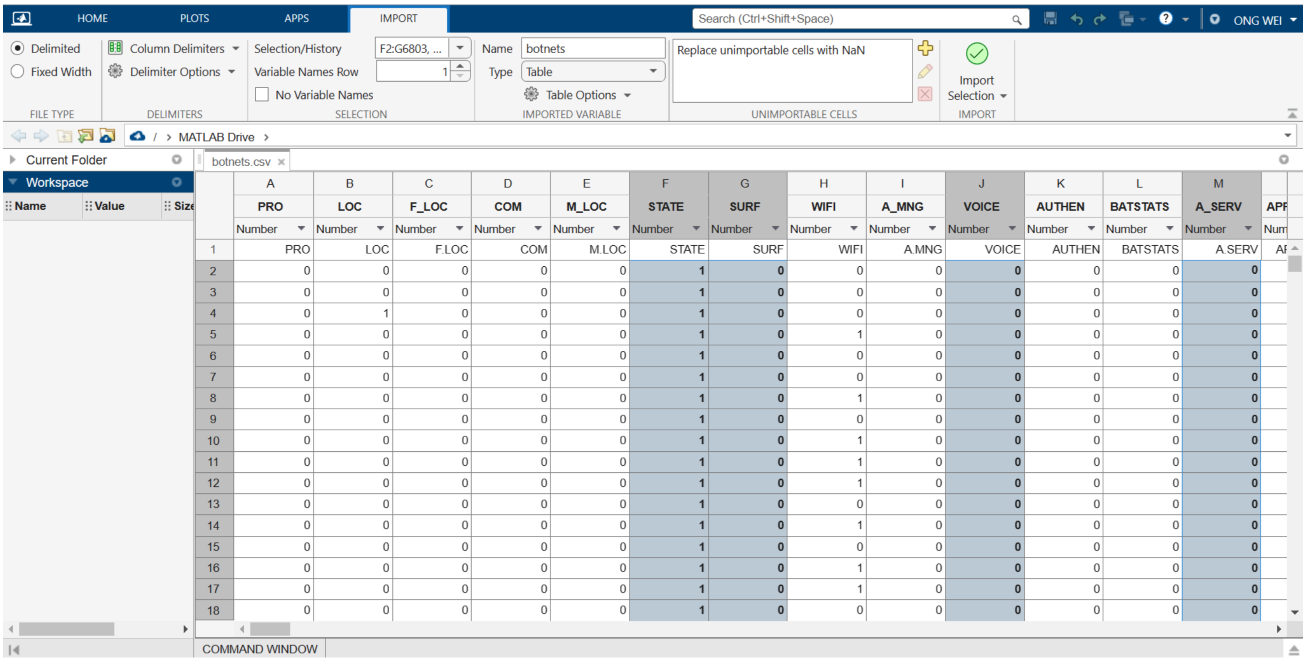


Figure 4.9 How features are selected

The figure above shows how the features are selected from the implemented data. We had to repeat the same step in order to produce the data of features based on the three different approaches.

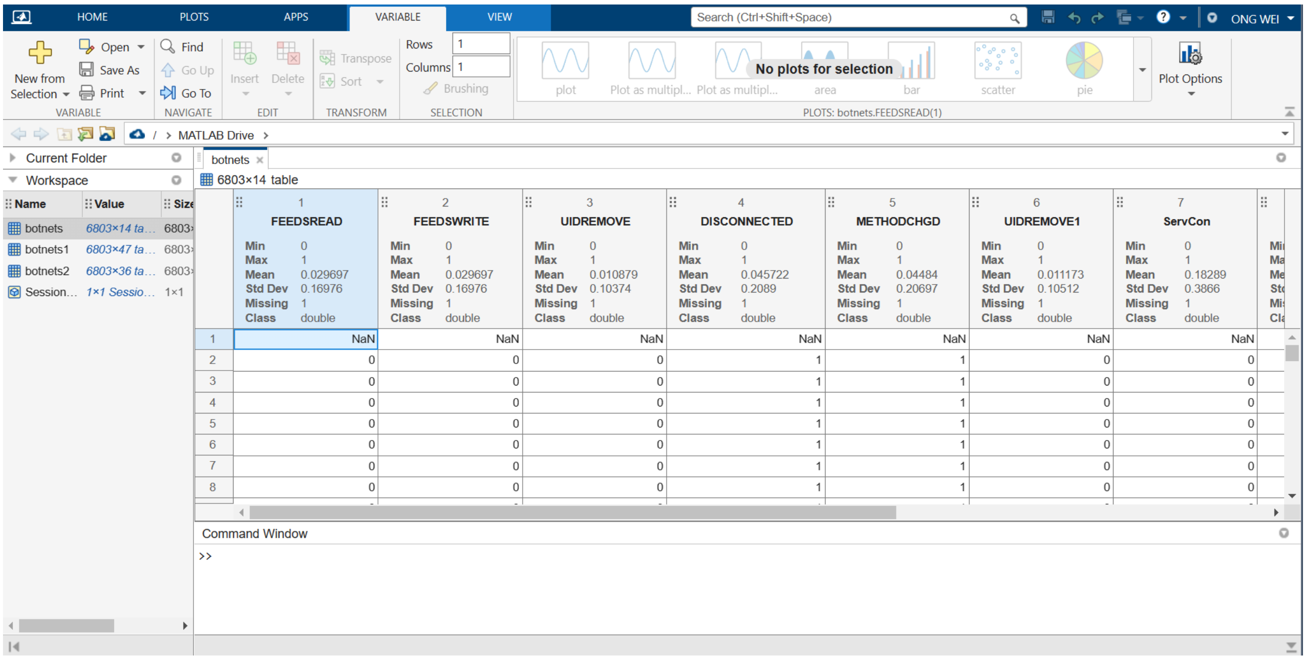
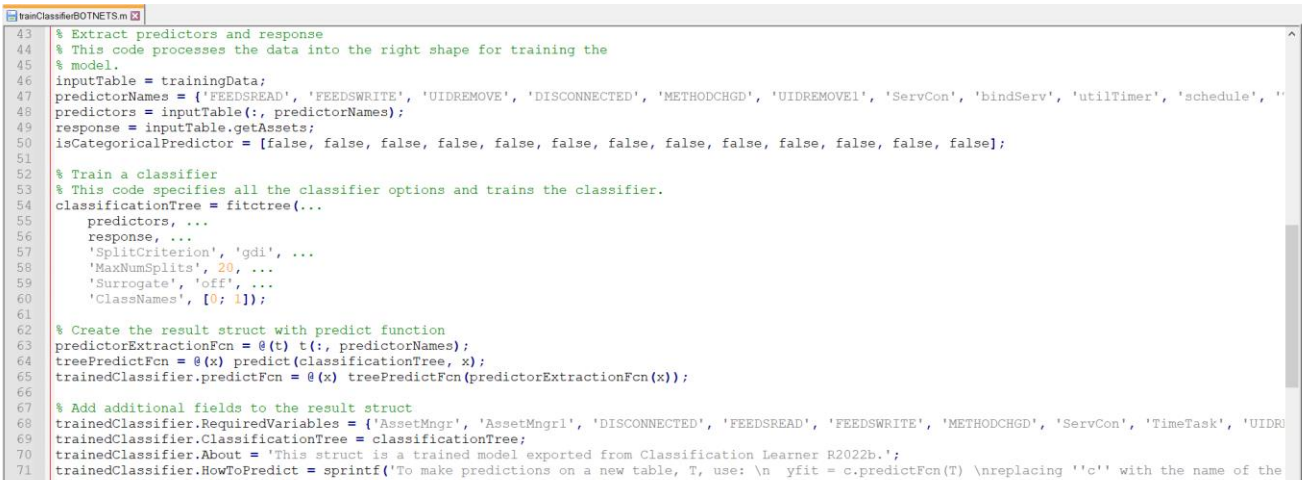


Figure 4.10 The features were selected

Figure 4.10 shows the features that were selected which form to 3 dataset which is botnet, botnet1, and botnet2.



```
% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'FEEDSREAD', 'FEEDSWRITE', 'UIDREMOVE', 'DISCONNECTED', 'METHODCHGD', 'UIDREMOVE1', 'ServCon', 'bindServ', 'utilTimer', 'schedule', '
predictors = inputTable(:, predictorNames);
response = inputTable.getAssets;
isCategoricalPredictor = [false, false, false, false, false, false, false, false, false, false, false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the classifier.
classificationTree = fitctree(...
    predictors, ...
    response, ...
    'SplitCriterion', 'gdi', ...
    'MaxNumSplits', 20, ...
    'Surrogate', 'off', ...
    'ClassNames', [0; 1]);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
treePredictFcn = @(x) predict(classificationTree, x);
trainedClassifier.predictFcn = @(x) treePredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.RequiredVariables = {'AssetMngr', 'AssetMngr1', 'DISCONNECTED', 'FEEDSREAD', 'FEEDSWRITE', 'METHODCHGD', 'ServCon', 'TimeTask', 'UIDR
trainedClassifier.ClassificationTree = classificationTree;
trainedClassifier.About = 'This struct is a trained model exported from Classification Learner R2022b.';
trainedClassifier.HowToPredict = sprintf('To make predictions on a new table, T, use: \n  yfit = c.predictFcn(T) \nreplacing ''c'' with the name of the
```

Figure 4.11 Coding of training using classifier of decision tree

Figure 4.11 shows the code has been developed to train the features using the machine learning classifier in MATLAB.

Machine learning is the study of algorithms that predict decisions based on samples of data using a computer without explicit programming. There are numerous classifiers available for machine learning, including decision tree, discriminate analysis, logistic regression, naive bayes, support vector machine, nearest neighbour, assemble, and neural network.

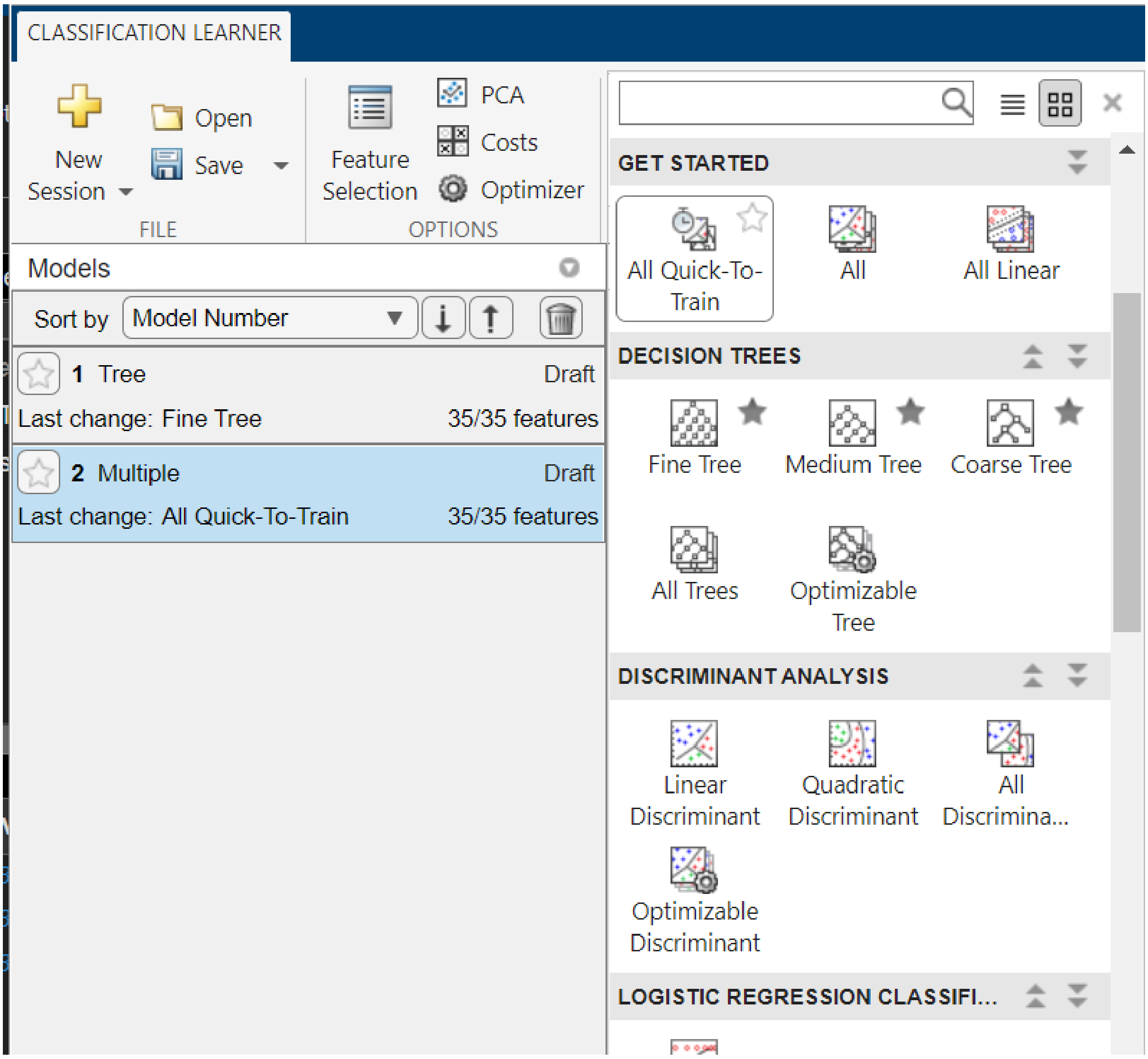


Figure 4.12 Select the classifier to train

In this study, we proposed to use all "quick to train" so that it can run all classifiers to get the highest accuracy. In addition, we also train the nearest neighbour classifier and support vector machine classifier.

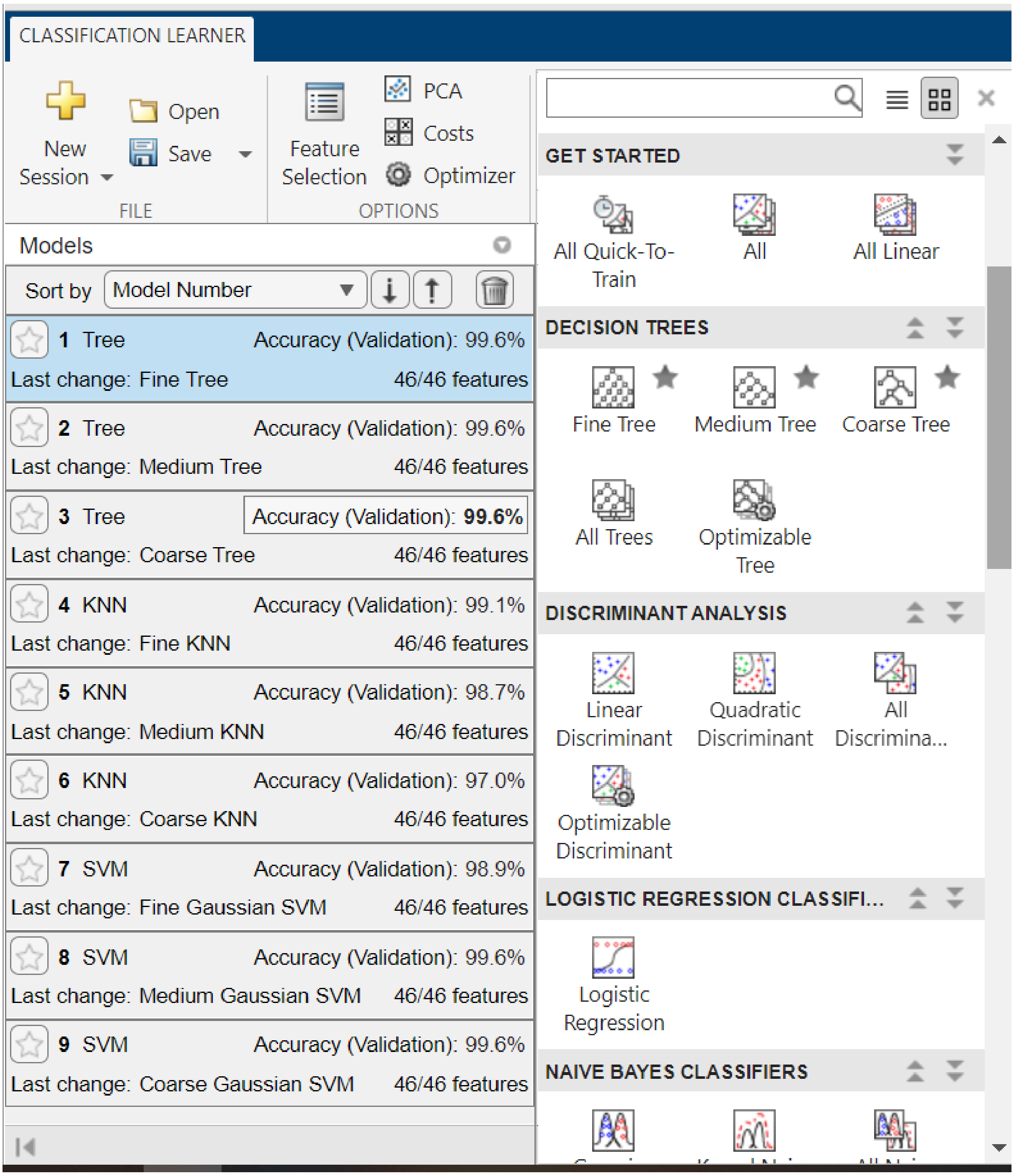


Figure 4.13 The accuracy of classifiers

Figure 4.13 shows the numbers of accuracy based on the trained classifier. The highest degree of precision will be detected. We chose to evaluate the precision of the three classifiers, decision tree, nearest neighbour, and support vector machines. Each classifier employs its own set of algorithms. Fine, Medium, and Coarse are used in decision trees and nearest neighbour (KNN), whereas support vector machines use Fine Gaussian, Medium Gaussian, and Coarse Gaussian (SVM).

4.3 Result

Table 4.1 Accuracy based on the classifier

Approach	Classifier	Fine	Medium	Coarse	Preferred Classifier
1st	Decision Tree	99.7%	99.7%	99.8%	✓
	KNN	99.8%	99.6%	98.7%	
	SVM	99.6%	99.8%	99.8%	
2nd	Decision Tree	99.6%	99.6%	99.6%	✓
	KNN	99.1%	98.9%	96.9%	
	SVM	99.0%	99.6%	99.6%	
3rd	Decision Tree	99.6%	99.6%	99.6%	✓
	KNN	99.2%	98.6%	96.9%	
	SVM	98.8%	99.6%	99.6%	

Table 4.1 above shows the accuracy based on three different classifiers which is decision tree, nearest neighbour (KNN), and support vector machines (SVM). In comparison to the other classifiers, the accuracy of the decision tree classifier is the highest based on the above result.

Table 4.2 Accuracy in Decision Tree Classifier

		Decision Tree Classifier		
Approach	Number Features	Fine	Medium	Coarse
1st	14	99.7	99.7	99.8
2nd	36	99.6	99.6	99.6
3rd	47	99.6	99.6	99.6
Preferred Algorithm				✓

Table 4.2 above shows the accuracy based on the decision tree algorithms. We are able to choose a suitable and trustworthy classifiers algorithm based on the aforementioned result. We select coarse tree classifiers as our results because it achieves the highest total accuracy when compared to fine and coarse tree classifiers.

Table 4.3 Feature of PMCC

Feature Selection Method	Product Moment Correlation Coefficient (PMCC)
Classifier	Coarse Tree
Highest accuracy	99.80%

The table above shows that this research used the product moment correlative coefficient (PMCC) as a feature selection method. The coarse tree algorithm has been selected as the best train classifier with the highest accuracy, 99.8 %.

Once the classifier algorithm has been selected, the next step is to determine the true positive rate (TPR) and false positive rate (FPR). The True Positive Rate (TPR) is the proportion of correct definite outcomes among all definite examples available during the test. On the other hand, the False Positive Rate (FPR) describes the frequency of false positives among all regrettable cases available throughout the test.

The Receiver Operating Characteristic (ROC) is an estimation of issue execution at different edge settings. AUC is the degree or percentage of distinguishability, whereas ROC is the likelihood curve. It indicates the model's suitability for class recognition. The better the model predicts 0s as 0s and 1s as 1s, the higher the AUC. According to the connection, the higher the AUC, the better the model recognizes the target with restrictions and no infection.

Figure 4.14 – 4.19 will show the result confusion matrix for three different approaches.

The first approach

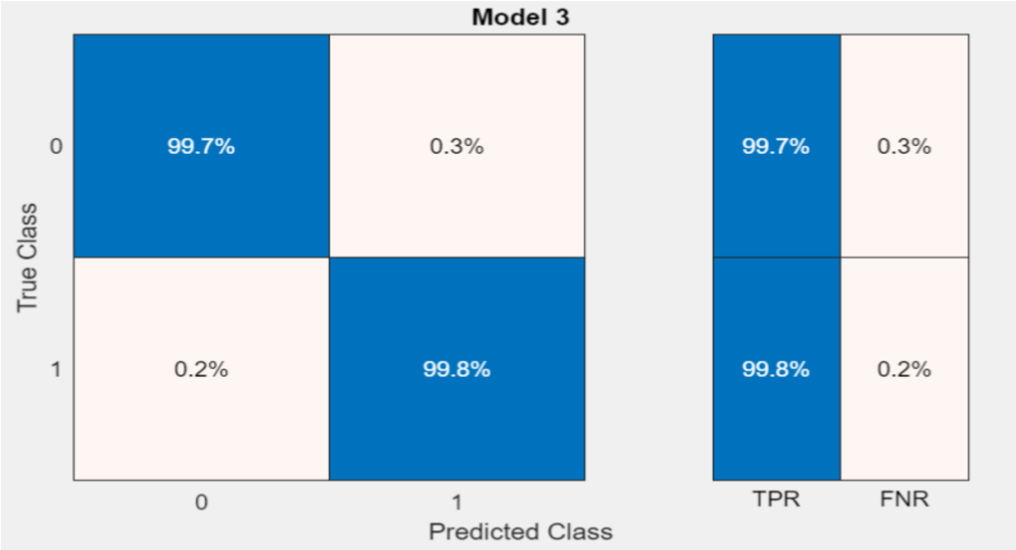


Figure 4.14 show the correlative matrix for the first approach

From the result of the first approach, we can identify the value of true positive (TP) is 99.7%, false negative (FN) is 0.3%, false positive (FP) is 0.2%, and true negative (TN) is 99.8%.

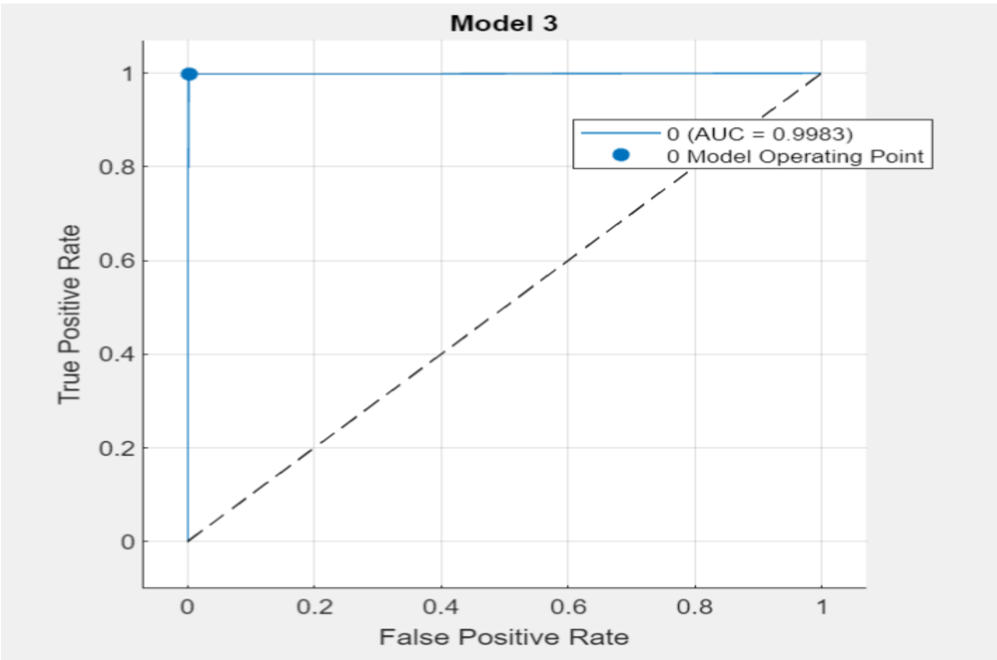


Figure 4.15 ROC Curve for the first approach

The result shows that AUC is 0.9983 which means the roc curve is reliable to be trusted and chosen.

The second approach

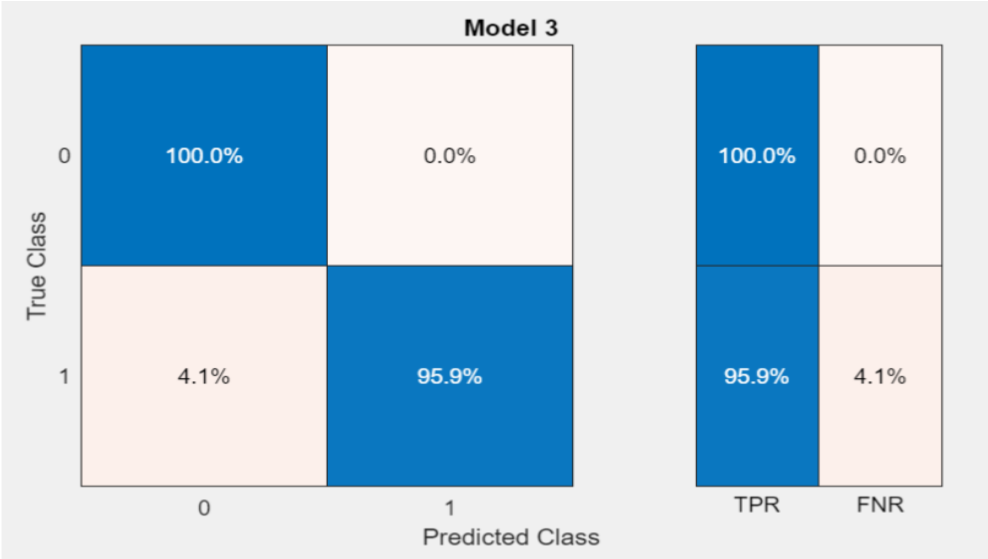


Figure 4.16 show the correlative matrix for the second approach

From the result of the second approach, we can identify the value of true positive (TP) is 100%, false negative (FN) is 0%, false positive (FP) is 4.1%, and true negative (TN) is 95.9%.

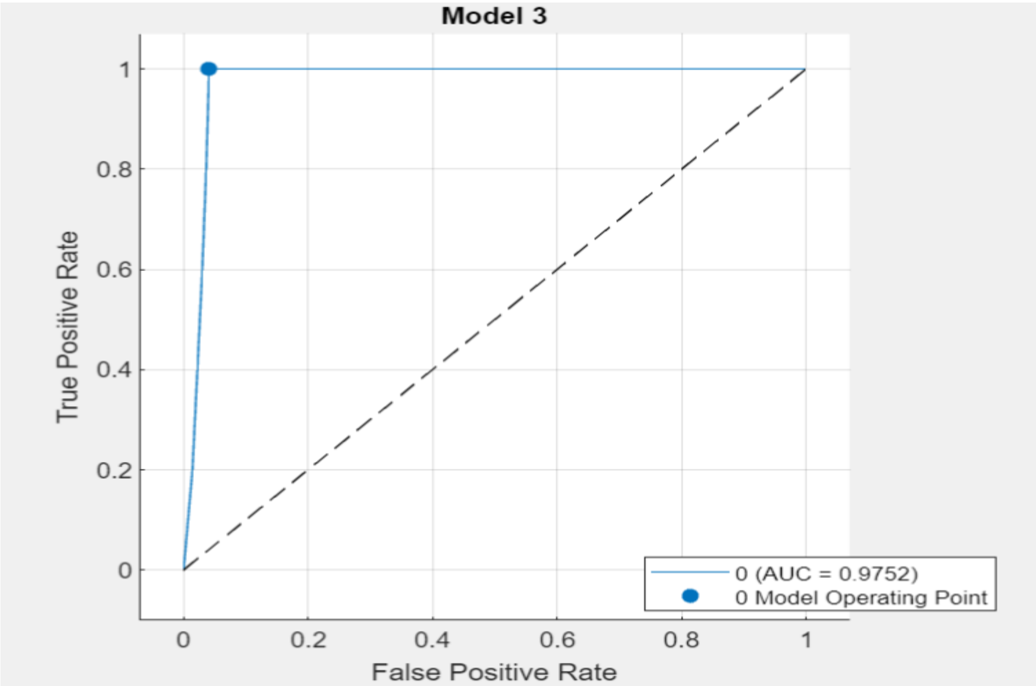


Figure 4.17 ROC Curve for the second approach

The result shows that AUC is 0.9752 which means the roc curve is reliable to be trusted and chosen.

The third approach

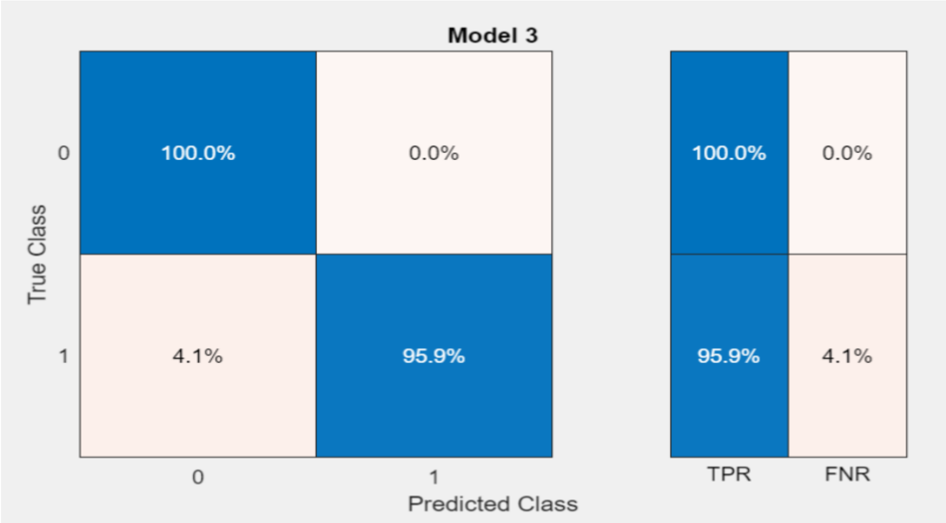


Figure 4.18 show the correlative matrix for the third approach

From the result of the third approach, we can identify the value of true positive (TP) is 100%, false negative (FN) is 0%, false positive (FP) is 4.1%, and true negative (TN) is 95.9%.

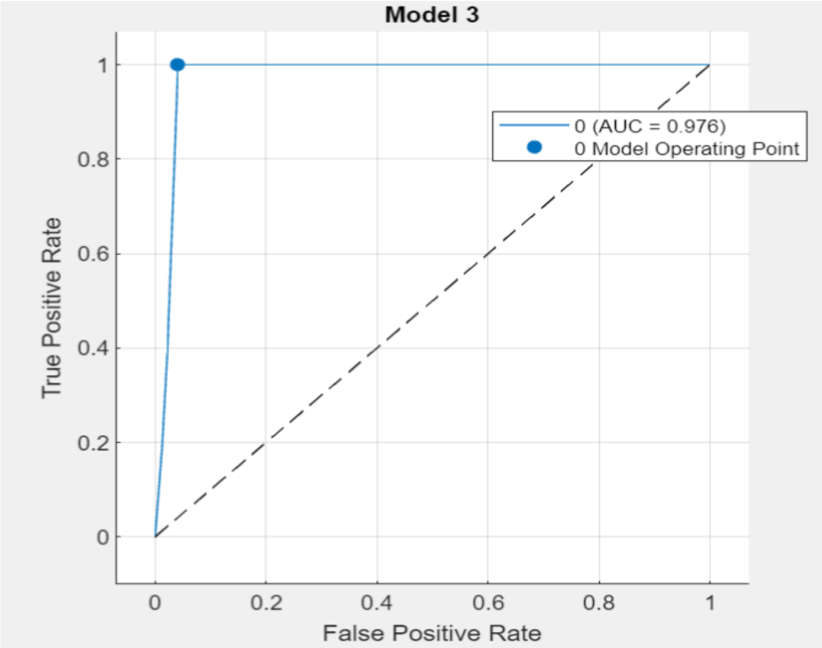


Figure 4.19 ROC Curve for the third approach

The result shows that AUC is 0.976 which means the roc curve is reliable to be trusted and chosen.

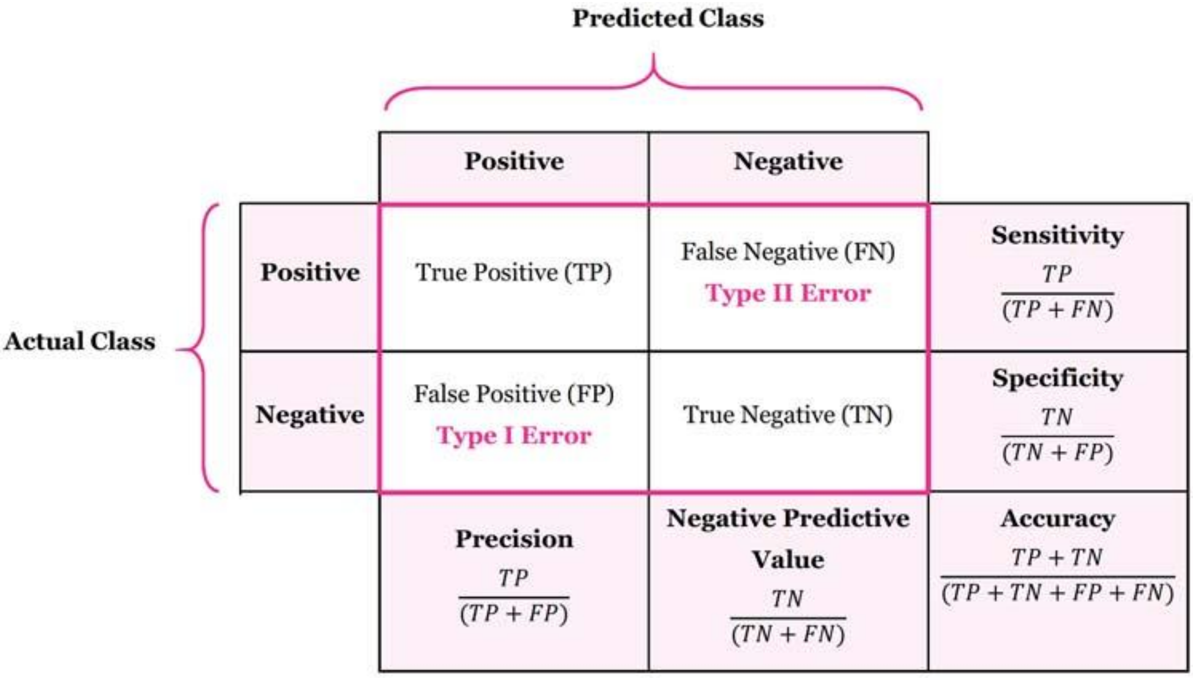


Figure 4.20 Confusion matrix table

True Positive (TP): It is the number of correctly classified instances as positive. It means that how successful a system is in detecting malware as malicious. As the true positive increases, the result is better.

False Positive (FP): It is the number of incorrectly classified instances as positive. It means that the ratio of which the algorithm considers normal data as malicious. As the false positive decreases, it shows that the system is more accurate.

True Negative (TN): It is the number of correctly classified instances as negative.

False Negative (FN): It is the number of incorrectly classified instances as negative.

Accuracy: It shows how accurately the system can detect malware.

Precision: It is the number of instances correctly classified as class X among those classified as class X.

Figure 4.20 above shows the confusion table used as a reference to count the value of precision. We also can get the value of sensitivity, specificity, accuracy, positive predictive value, and negative predictive value.

Table 4.4 Result and finding.

Classifier Algorithm	Number Features	ACC	TPR	FPR	AUC
Coarse Tree	14f	99.8	0.998	0.003	0.9983
	36f	99.6	1	0.041	0.9752
	47f	99.6	1	0.041	0.976

4.4 Development

BotnetModelBuilding.py

```
BotnetModelBuild.py
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
botnet = pd.read_csv('FeatureSelected.csv')

df = botnet.copy()
target = 'Result'
encode = ['SUBSCRIBED_FEEDS_READ','SUBSCRIBED_FEEDS_WRITE','android_intent_action_UID_REMOVED','UMS_DISCONNECTED','INPUT_METHOD_CHANGED','UID_REM


# # Separating X and y
X = df.drop('Result', axis=1)
Y = df['Result']

# Build decision tree model
clf = DecisionTreeClassifier()
clf.fit(X, Y)

# Saving the model
import pickle
pickle.dump(clf, open('botnet_clf.pkl', 'wb'))

```

Figure 4.21 Code to build the model

Figure 4.21 above shows the coding of the model building. There are 3 python library was imported into the file. First is pandas which use to analyse CSV file. Following that is the Decision Tree Classifier which allows us to use the decision tree classifier to train our model. Lastly, the pickle is used to save the trained model.

NavPrediction.py

```
NavPrediction.py
import streamlit as st
import pandas as pd
import numpy as np
import pickle
import random
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image
from streamlit_option_menu import option_menu

EXAMPLE_NO = 1
st.set_page_config(layout='wide', initial_sidebar_state='expanded')

def streamlit_menu(example=1):
    if example == 1:
    # horizontal menu w/o custom style
        selected = option_menu(
            menu_title=None,  # required
            options=["Home", "Heatmap Analysis", "User Guide"],  # required
            icons=["house", "book", "envelope"],  # optional
            menu_icon="cast",  # optional
            default_index=0,  # optional
            orientation="horizontal",
        )
        return selected

selected = streamlit_menu(example=EXAMPLE_NO)

botnet = pd.read_csv('FeatureSelected.csv')

```

Figure 4.22 Code for the Prediction System

Figure 4.22 shows the coding for the prediction system. There are 9 python library was imported into the system. A detail description of each library will explain at table below.

Table Python Library and Description

Python Library	Description
Streamlit	Python language app framework
Pandas	Use to analyse csv file
Numpy	Working with arrays
Pickle	Read saved model
Random	To generate random number
Matplotlib.pyplot	A plotting library MATLAB
Seaborn	To make statistical graphics
Image	To allow python to interpret the image
Option menu	A simple Streamlit component that allows users to select a single item from a list of options in a menu

```
def convert_df(df):
# IMPORTANT: Cache the conversion to prevent computation on every rerun
    return df.to_csv(index=False).encode('utf-8')
# data of Player and their performance
data = {'SUBSCRIBED_FEEDS_READ': [random.randint(0,1)],
        'SUBSCRIBED_FEEDS_WRITE': [random.randint(0,1)],
        'android_intent_action_UID_REMOVED': [random.randint(0,1)],
        'UMS_DISCONNECTED': [random.randint(0,1)],
        'INPUT_METHOD_CHANGED': [random.randint(0,1)],
        'UID_REMOVED': [random.randint(0,1)],
        'ServiceConnection': [random.randint(0,1)],
        'bindService': [random.randint(0,1)],
        'Ljava_util_Timer_schedule': [random.randint(0,1)],
        'Ljava_util_TimerTask': [random.randint(0,1)],
        'Ljava_util_Date': [random.randint(0,1)],
        'AssetManager': [random.randint(0,1)],
        'Landroid_content_res_AssetManager': [random.randint(0,1)],
        'getAssets': [random.randint(0,1)]}
```

Figure 4.23 Code for generating a random dataset file.

Figure 4.23 shows the code to generate some random dataset files with these 14 features to allow the user to download it and go the prediction in the system.

```
if uploaded_file is not None:
        input_df = pd.read_csv(uploaded_file)
else:
        st.sidebar.subheader('User Input Features')
        def user_input_features():
            SUBSCRIBED_FEEDS_READ = st.sidebar.slider('SUBSCRIBED_FEEDS_READ', 0, 1, 0)
            SUBSCRIBED_FEEDS_WRITE = st.sidebar.slider('SUBSCRIBED_FEEDS_WRITE', 0, 1, 0)
            android_intent_action_UID_REMOVED = st.sidebar.slider('android_intent_action_UID_REM
            UMS_DISCONNECTED = st.sidebar.slider('UMS_DISCONNECTED', 0, 1, 0)
            INPUT_METHOD_CHANGED = st.sidebar.slider('INPUT_METHOD_CHANGED', 0, 1, 0)
            UID_REMOVED = st.sidebar.slider('UID_REMOVED', 0, 1, 0)
            ServiceConnection = st.sidebar.slider('ServiceConnection', 0, 1, 0)
            bindService = st.sidebar.slider('bindService', 0, 1, 0)
            Ljava_util_Timer_schedule = st.sidebar.slider('Ljava_util_Timer_schedule', 0, 1, 0)
            Ljava_util_TimerTask = st.sidebar.slider('Ljava_util_TimerTask', 0, 1, 0)
            Ljava_util_Date = st.sidebar.slider('Ljava_util_Date', 0, 1, 0)
            AssetManager = st.sidebar.slider('AssetManager', 0, 1, 0)
            Landroid_content_res_AssetManager = st.sidebar.slider('Landroid_content_res_AssetMan
            getAssets = st.sidebar.slider('getAssets', 0, 1, 0)

            data = {'SUBSCRIBED_FEEDS_READ': SUBSCRIBED_FEEDS_READ,
                    'SUBSCRIBED_FEEDS_WRITE': SUBSCRIBED_FEEDS_WRITE,
                    'android_intent_action_UID_REMOVED': android_intent_action_UID_REMOVED,
                    'UMS_DISCONNECTED': UMS_DISCONNECTED,
                    'INPUT_METHOD_CHANGED': INPUT_METHOD_CHANGED,
                    'UID_REMOVED': UID_REMOVED,
                    'ServiceConnection': ServiceConnection,
                    'bindService': bindService,
                    'Ljava_util_Timer_schedule': Ljava_util_Timer_schedule,
                    'Ljava_util_TimerTask': Ljava_util_TimerTask,
                    'Ljava_util_Date': Ljava_util_Date,
                    'AssetManager': AssetManager,
                    'Landroid_content_res_AssetManager': Landroid_content_res_AssetManager,
                    'getAssets': getAssets}
            features = pd.DataFrame(data, index=[0])
            return features
        input_df = user_input_features()
```

Figure 4.24 Code of user input features value using sidebar.

Figure 4.24 shows the coding of the user input features value using the sidebar.

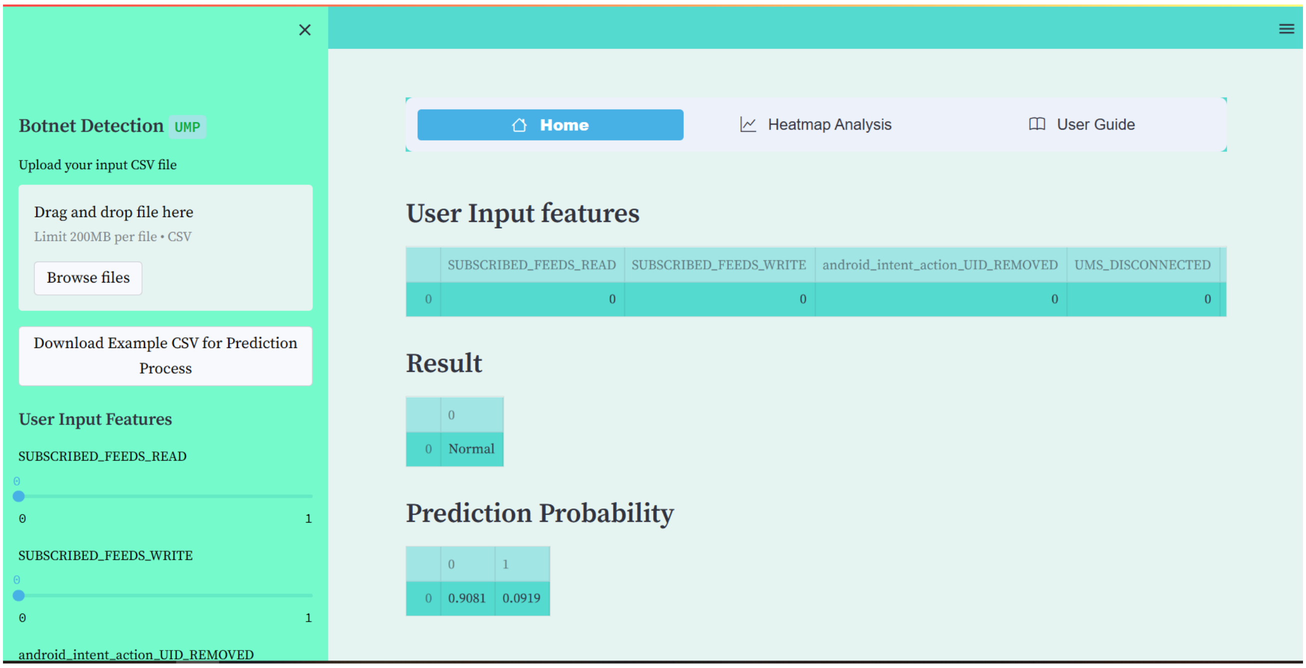


Figure 4.25 Interface of the Botnet Detection System.

On the left side of the page, there 2 buttons and 14 progress bars. The first button from the top is the browse files button which allows the user to upload the file to do the detection. The second button is the button that will download a random dataset CSV file for the user to upload the file for detection. On the right side of the page, There is a navigation bar with 3 column which is Home . Heatmap Analysis and User Guide. In the home part, the first section is the User Input Features which is it will show the input value of each feature. The second section which is the result of the detection and the last section is the prediction probability which shows how many percentages the input value is malware.

4.5 Web Hosting

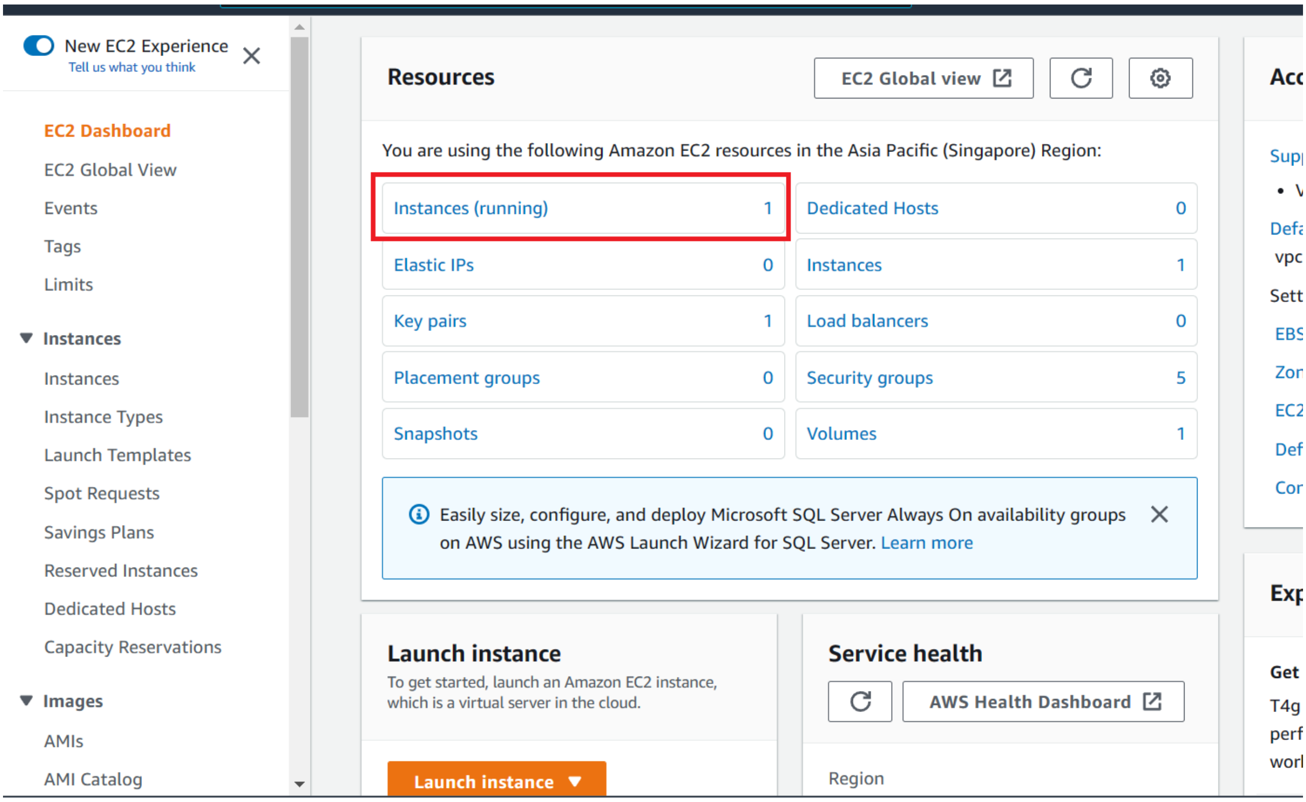


Figure 4.26 AWS EC2 Dashboard

Figure 4.26 shows the AWS EC2 dashboard. Click on the Instances (running) it will show the interface as Figure 4.27 at below.

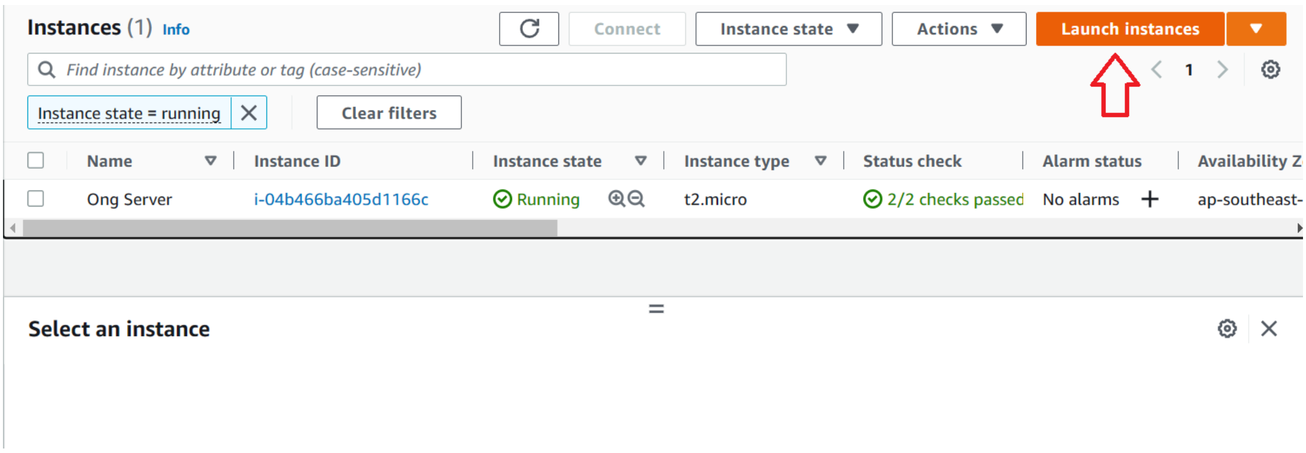


Figure 4.27 List of Instance

In Figure 4.27, it will show the current instance that my account has as a list. TO create a new instance To create a new instance, click on the orange button (Launch instances).

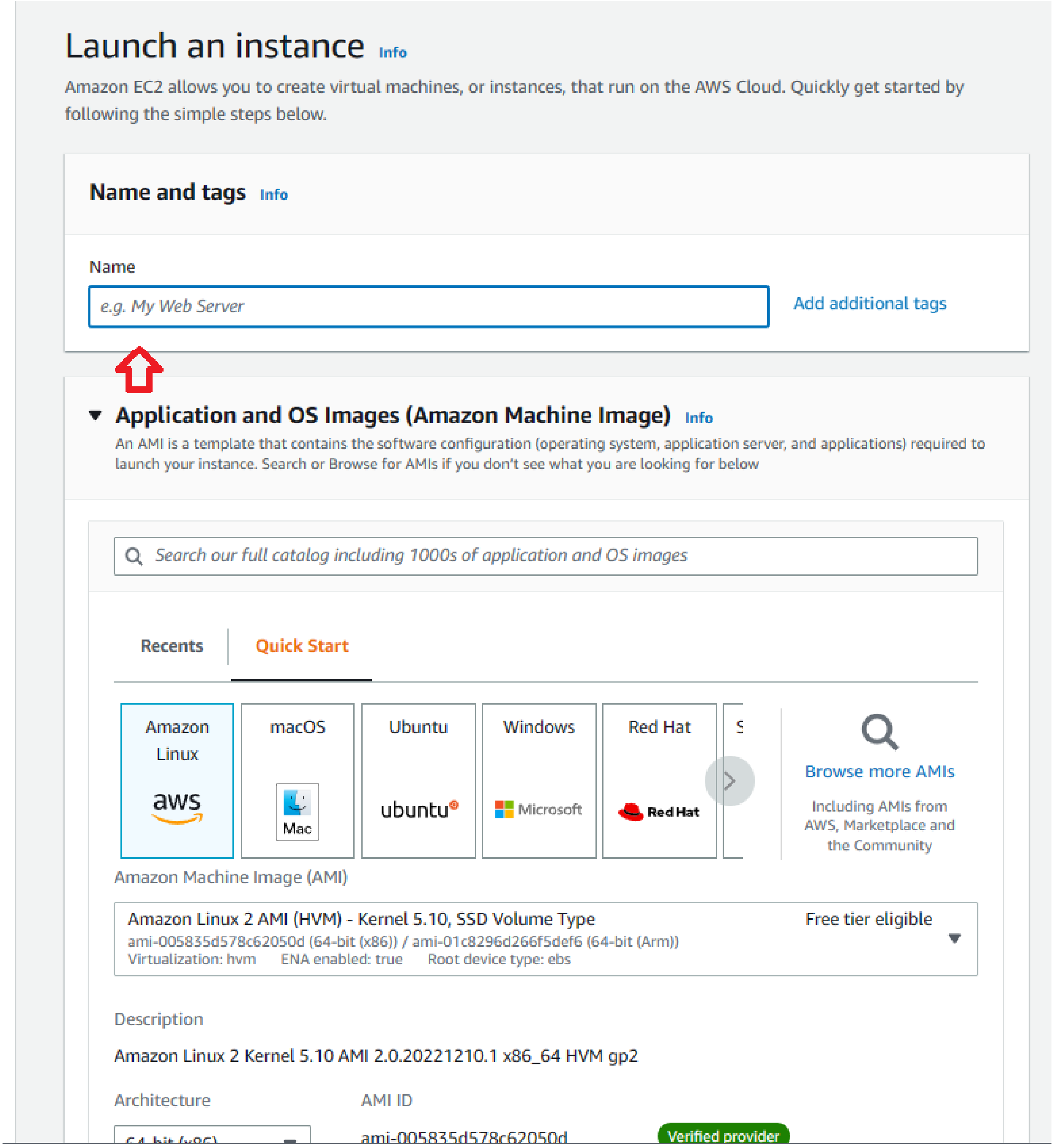


Figure 4.28 Name the instance

In Figure 4.28, put a name for the web server.

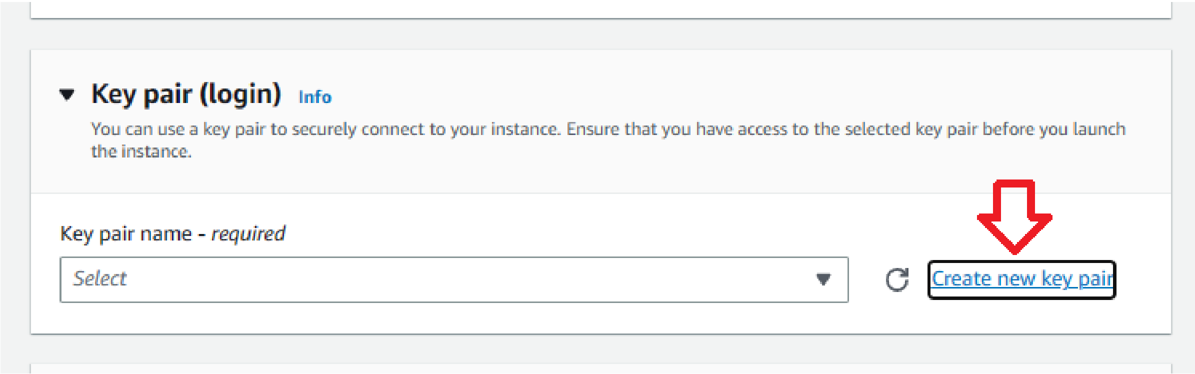


Figure 4.29 Create new key pair

Figure 4.29 shows how to create a new key pair for the web server. Key pair is needed because it is allow us to access our own AWS server ubuntu command line later. A .pem file will created and will be use later.

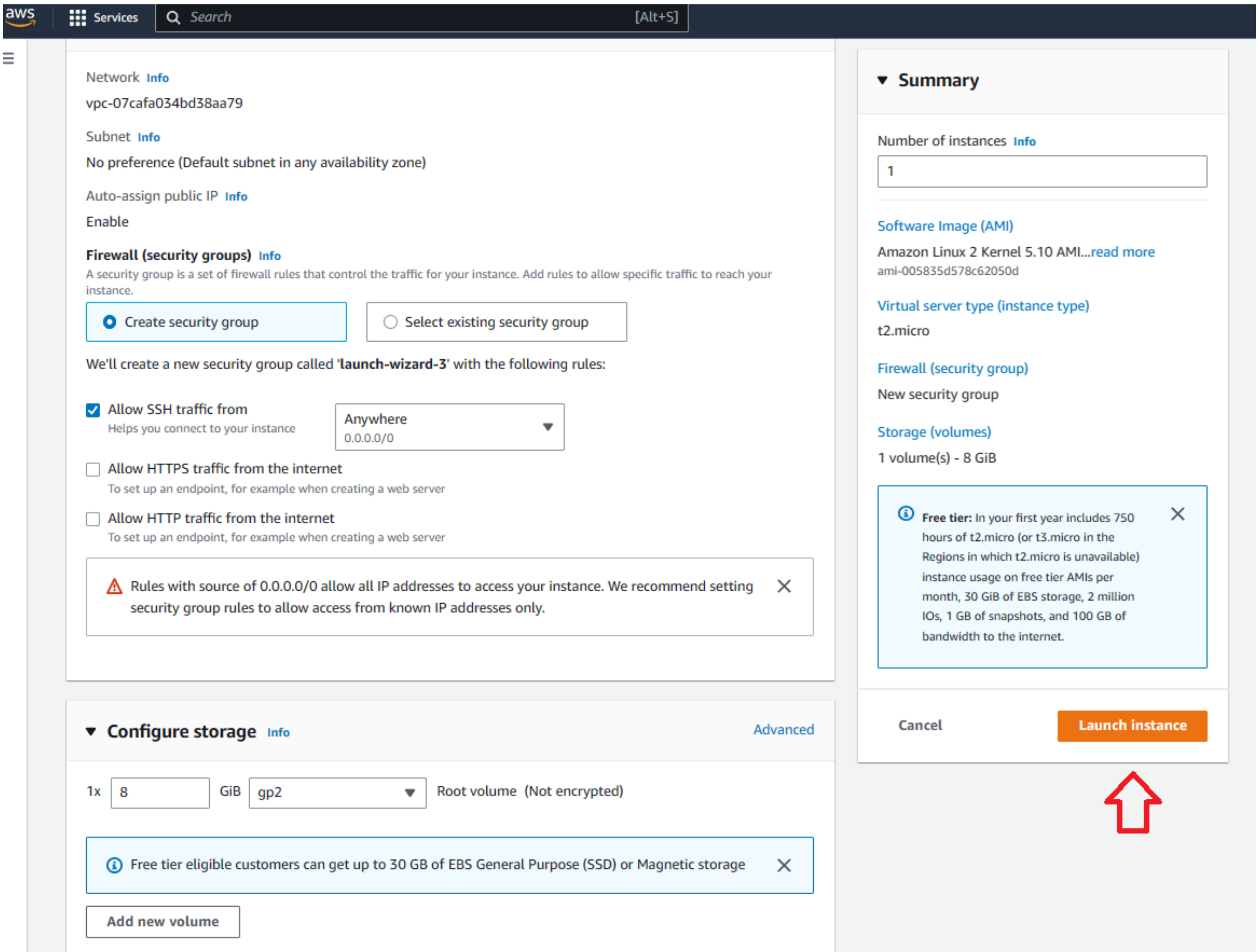


Figure 4.30 Other Settings

Figure 4.30 shows the continuous setting. We remain it as default and click the orange button (Lauch instance).

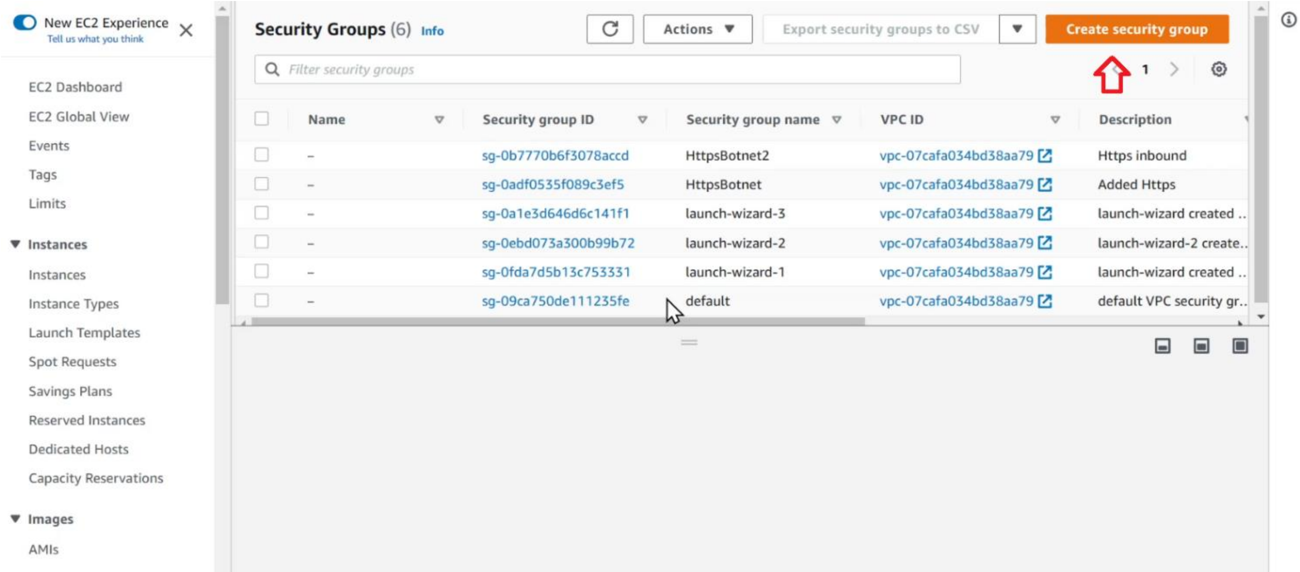


Figure 4.31 Security Group

Figure 4.31 shows the list of the security group. We can edit the security group by click the security group ID. As a demonstration we go to create security group.

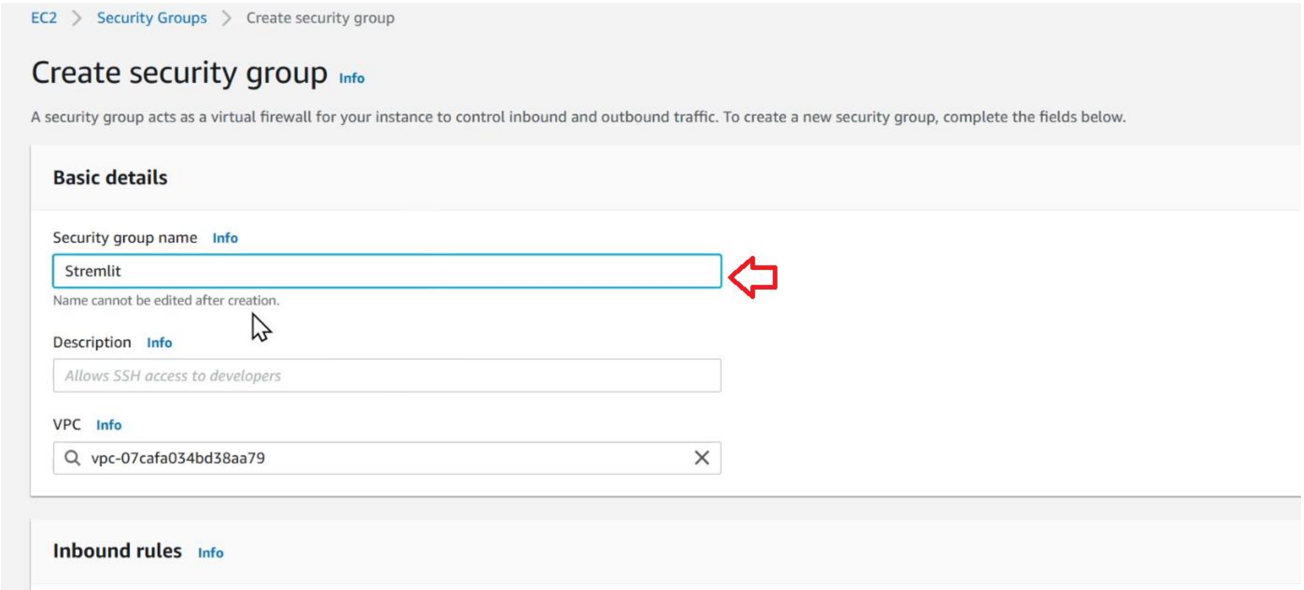


Figure 4.32 Create New Security Group

Figure 4.32 shows the information to create the security group. A security group name is needed because we can use it for different instance and select the correct VPC instance.

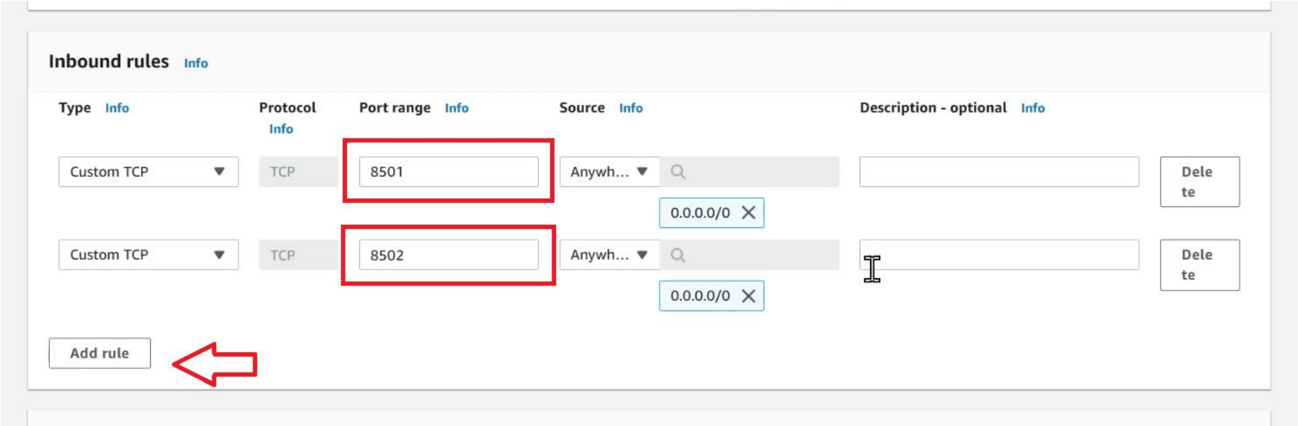


Figure 4.33 Update Inbound Rules

Figure 4.33 shows the information to update for inbound rules. The inbound rules is help the web server to filter incoming traffic. By default the inbound rules is empty. For our streamlit web application, it use the TCP with 8501 port and 8502 port. Therefore we need to configure the port range and set the source from anywhere which to allow user to access the web application.

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAh9+LroICiv37sqA0HQNj/TwvEt20TegLnVHrBzZWc/evB9ut
sGUNLtsEDiMs8ZCQd/QlVeRQ5TUE4YHrSnlbn3rOKpnUhyWqC+9st9BmhFf9Nf7k
0WlA2m4771ZI83QhJ9zdB3tRojzjI0GaZCUQXCQMIBACnKr64ZR8AqITyrTFAOld
yy73didlBib8pcbhTFQFDq4v8IQ0JcEDSgHlNUeDMlYQXpOr6DdKn4CF9AzBWuUX
0n+1hWzfU7oR/pteUSC3oJYorQwcGPnbyWSLrKEA1NkXWUQr4ZqYCWiXvgaU6VCa
QrIfQ0gQVcdnQAqpunEl3Yw5wJ79yTlYPmz4uQIDAQABAoIBAEQaf3fzwHSMg6lv
9U8JkQewL+Qj7ikSgyfSlJxj1wd/gWLN8Iw2ylnO+4Reizt1e4Q0grY/n3CTl6O0
rAwDMjKIqmfd/RHUhhw/YN3tfkUdmVSEM4rnV9NkZ3Q6aoxki+3gHYWPgUZxgGP+
kPVbQoz8oHs9qyF97gw9kb78IDF8mJCHUb0gmVSkQmGBf7+4Cp6TmsCORjSKQ0oY
xRBgPgqw6KgdvgVU1qUfUPJQPIJGhmNHRNi0W3tturF/Wi9WinvbkAJSeOXQwYPw
msYfclPMIG+auhd/uVwYWxtsv7DsfkMlbPJvWINc52sUuRqSOWoMXnXZuC4fP5dx
s6p3u1ECgYEAyyLjQ60frZO5lR+XMonm9gBAvOzCGqOXklAbUyMXp6LhrPskfDQ5
Yjje0ln74GmvfzhVFm5OSD/x7em93VG4UL/TiYTj9bsDjov3gyKYgdTvEpHQ/AXx
cV8+fdYXgnsZamot2fCGDSzgi3ZD5I8ecSmpdX6PL0yEE8QJ9JF4Kz0CgYEAqzuJ
KKX4m1JpxGpMqhO1gXXQa8kKRqwNf2C7jf49XiHtG+sI+ZYGbsvwc/T9PZUgYVU2
o7GKEs0oqT00c7ec0F86xgXLGMAI4q2/Y5hqsopz5wwihTfceZ/IDXvNgEvZJsoz
30zY2tnw8lhEz/9p5RES77xziWgVqepdxhpUyy0CgYEAl6fURBJcNBq10r9jAjgy
VjaaHnIj9/9qibGEtOzeHClekuZsts3Gia4rrJ/BjCla/H/yBm0TxJz44cZAGZuJ
H8AXDfRIvIyCe0nD4ANUGJoAYry6aW2GdD3HSessYh3FO81JrgwECJIYkgYZaenv
sEyKV03FWGqnsJoLVKvGK4ECgYBjneRCqKQEQKLVqP3m6EZtBYx2WGRJDylfFHio
t+Md02DtIAR9p0EgzjaaT0nQvOQ0m+2It+3a2E4yfy/3rjdlpjE8KPyp7nZ84ZX5
rB9OtWuOXCntFL2IaGNrLL42SGoRvgsFeuCiGa6qXp3R4AbMQ+2fWSRiKRKJYpah
fIoeFQKBgGRv1yjvlvyimC3ySCHubpkRHMHVmJCaDH2uAOC+dkuuvTHgcj+ZOtrf
1THtOLEQoiQJOZTQDYIFGk0Pe4mPS+JhFiaF5kp0DFRhyBbMVVpL4tOqyqpItYtO
1YGD3mRNRyNjKI5Ag5hF/xyV2l6JGzR2g23itEk/u6sgdbHd0/G8
-----END RSA PRIVATE KEY-----
```


Figure 4.34 RSA Key Pair

Figure 4.34 shows the .pem file which contain the RSA Private Key. This pair key will be use to access the AWS server.

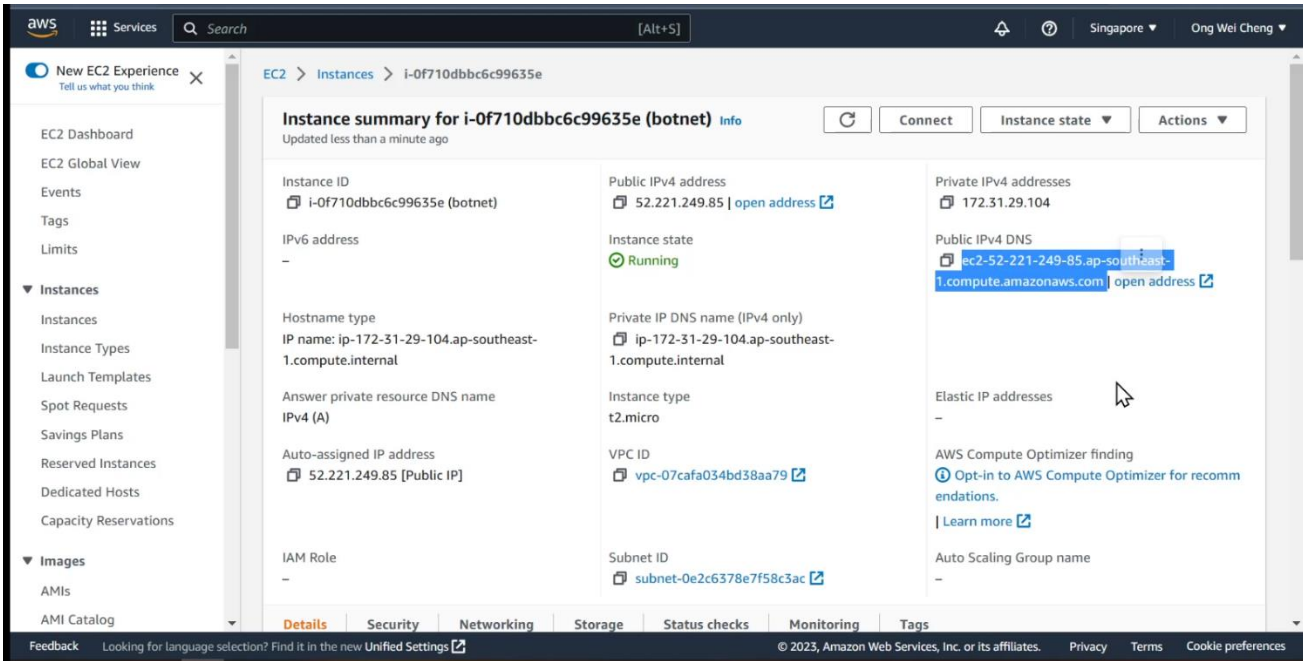


Figure 4.35 DNS Public IP

Figure 4.35 shows the Domain Name Server public ip for the AWS server. This DNS public ip will be use to access the AWS server.

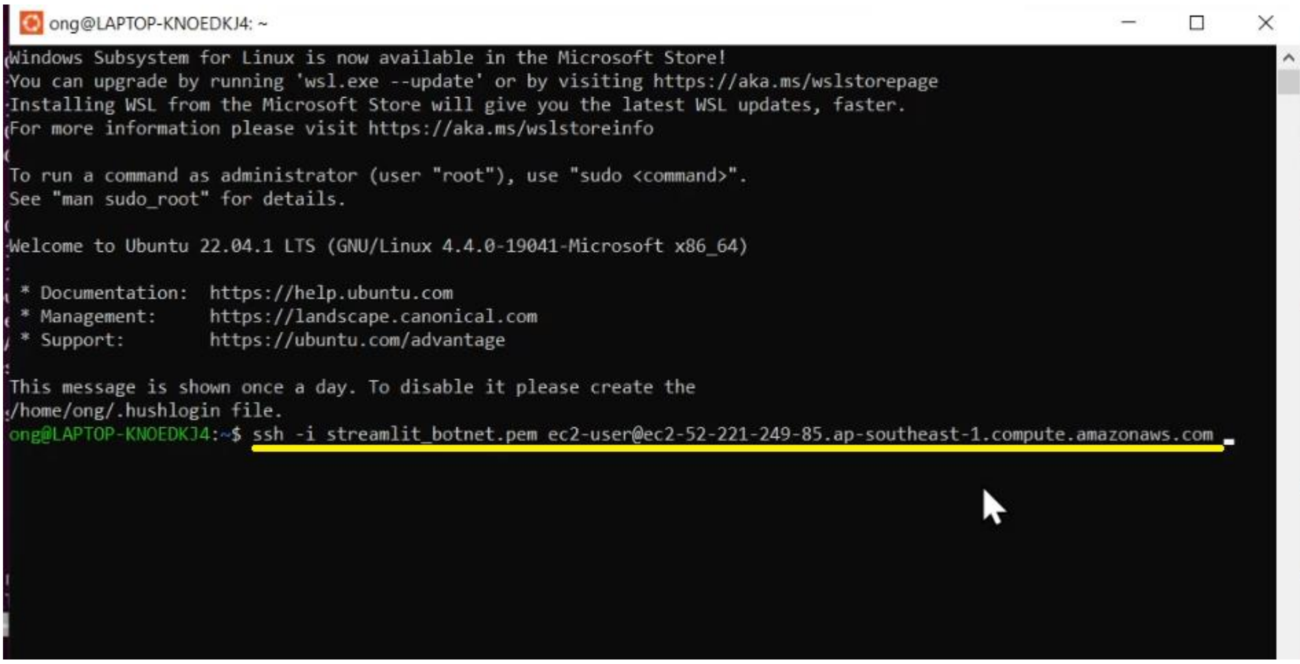


Figure 4.36 Command line to access the AWS server.

Figure 4.36 shows the command line to access the AWS server. The command is shh -i (the name of .pem file shows in Figure 4.34) ec2-user@(the DNS public ip in Figure 4.35).

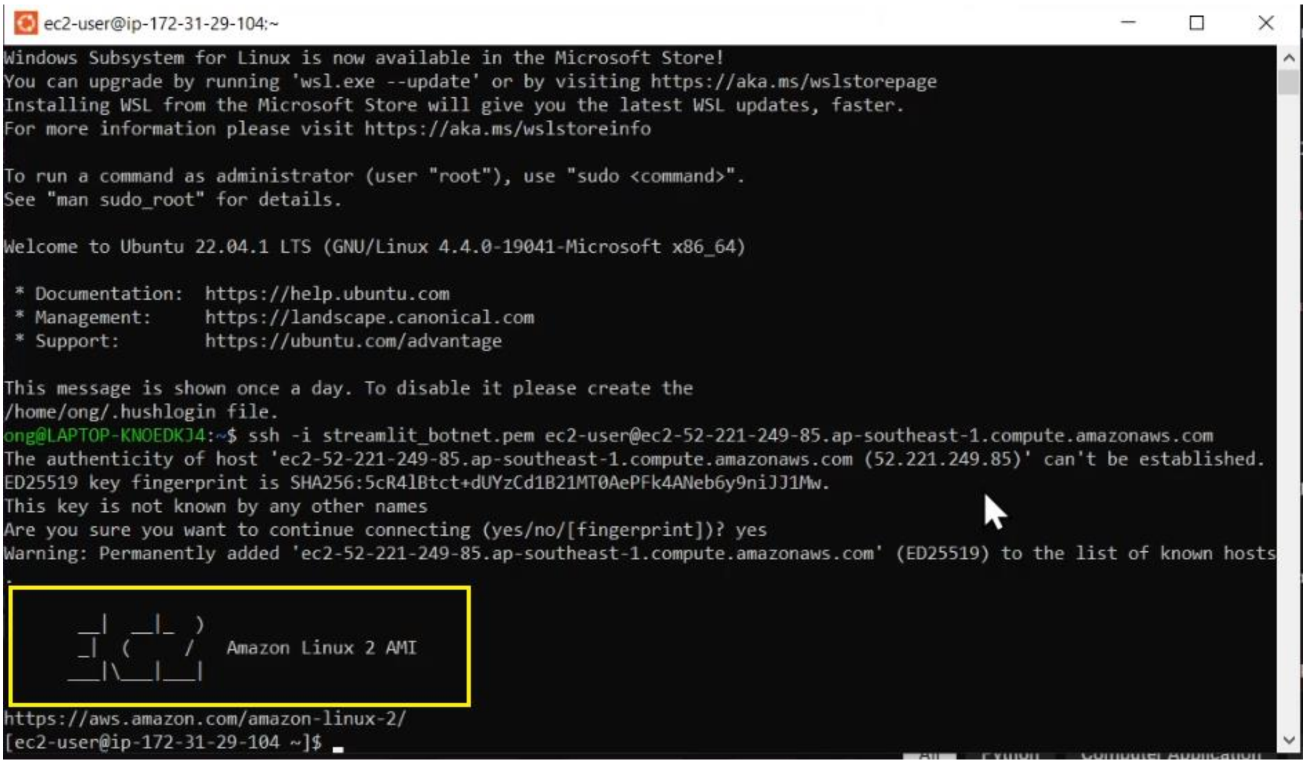


Figure 4.37 EC2 AWS server .

Figure 4.37 shows we are successful to access the EC2 AWS server.

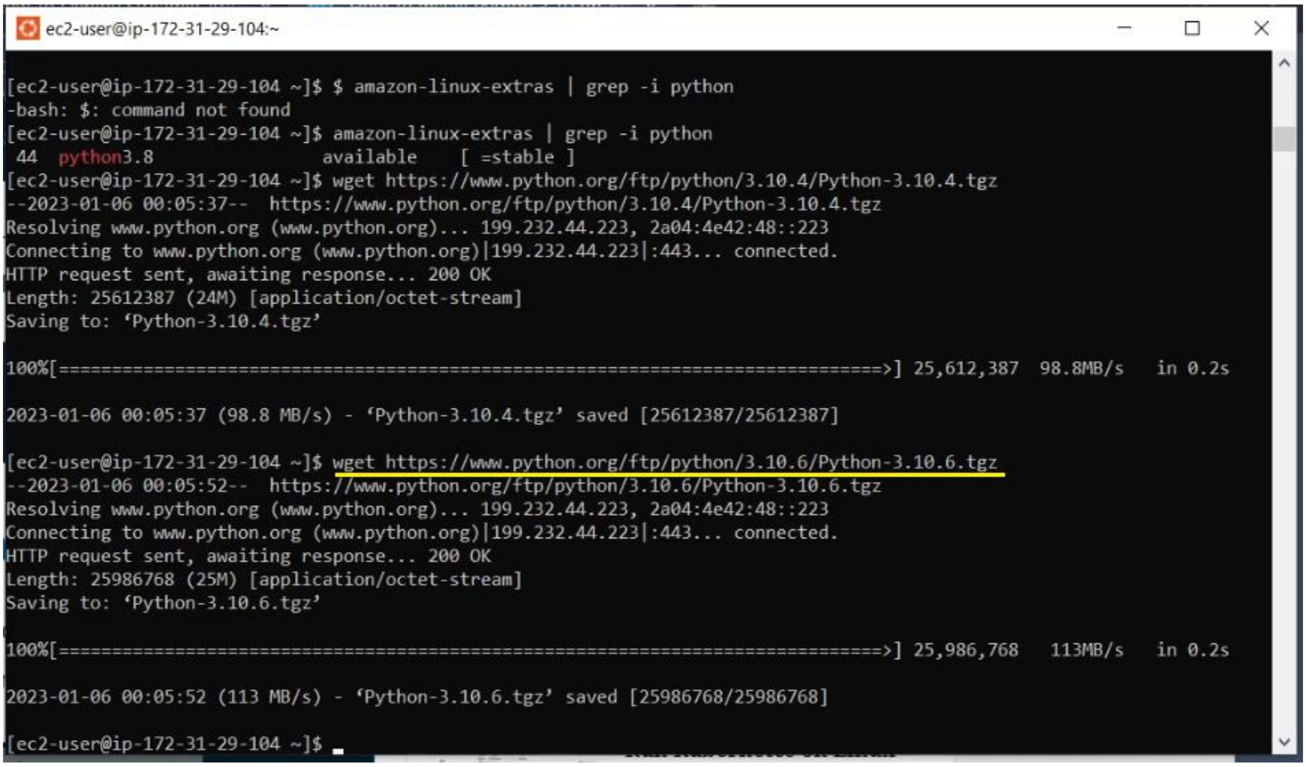


Figure 4.38 Install pyhton in the EC2 AWS Server

Figure 4.38 shows the python installation in the EC2 AWS Server.

Figure 4.39 Python validation in the EC2 AWS Server

Figure 4.39 shows how I validate the python is successful install in the EC2 AWS Server.

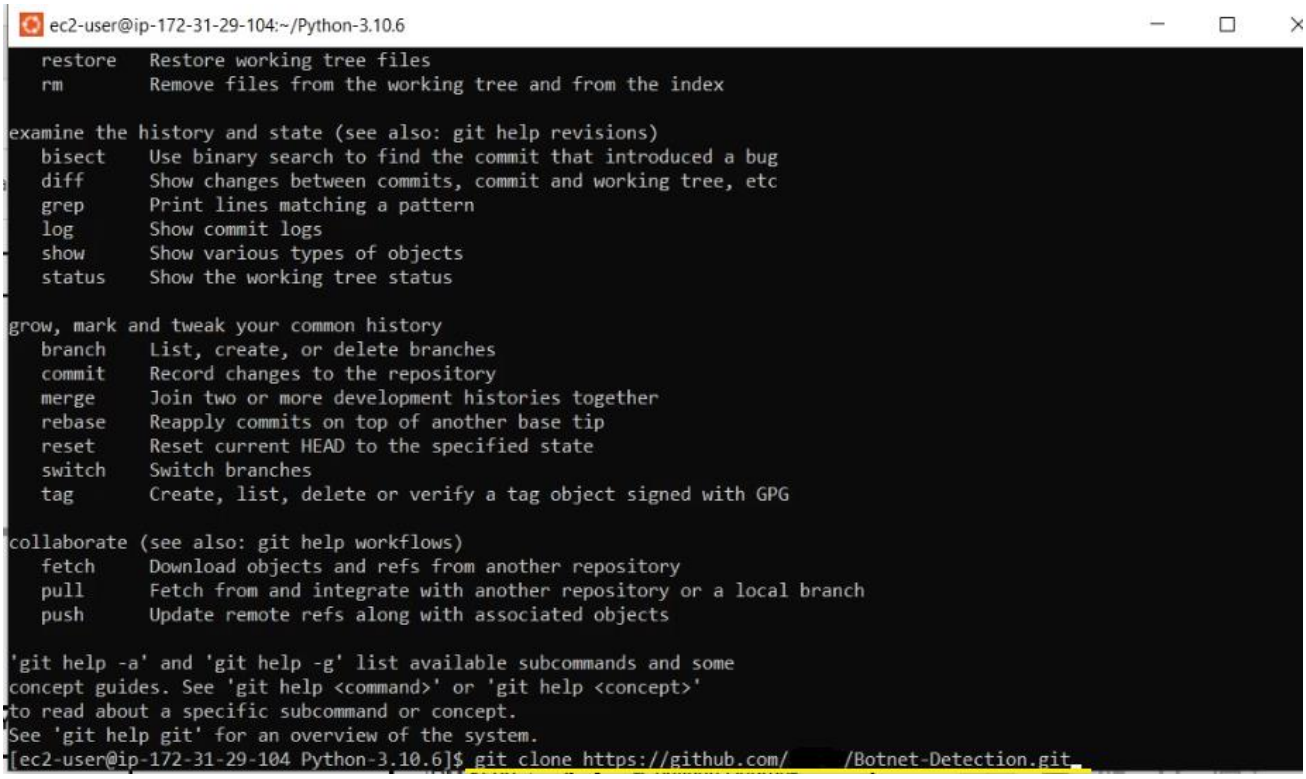


Figure 4.40 Git Clone My Repository

Figure 4.40 shows the cloning of my GitHub repository.

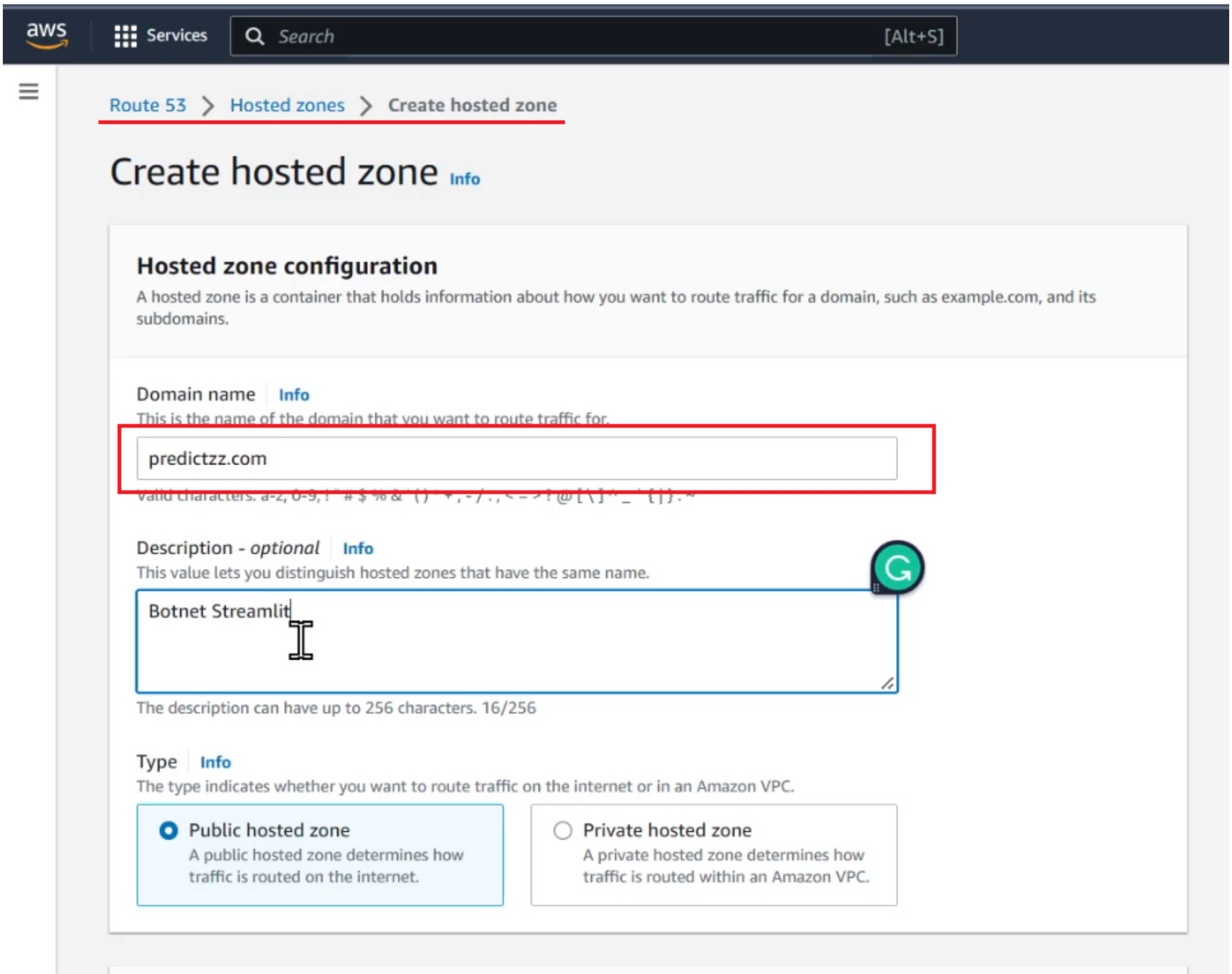


Figure 4.41 Set Domain Name in Route 53

In Figure 4.41, we need to put the correct domain name that we purchased and remain default setting for others setting and create the host zone.

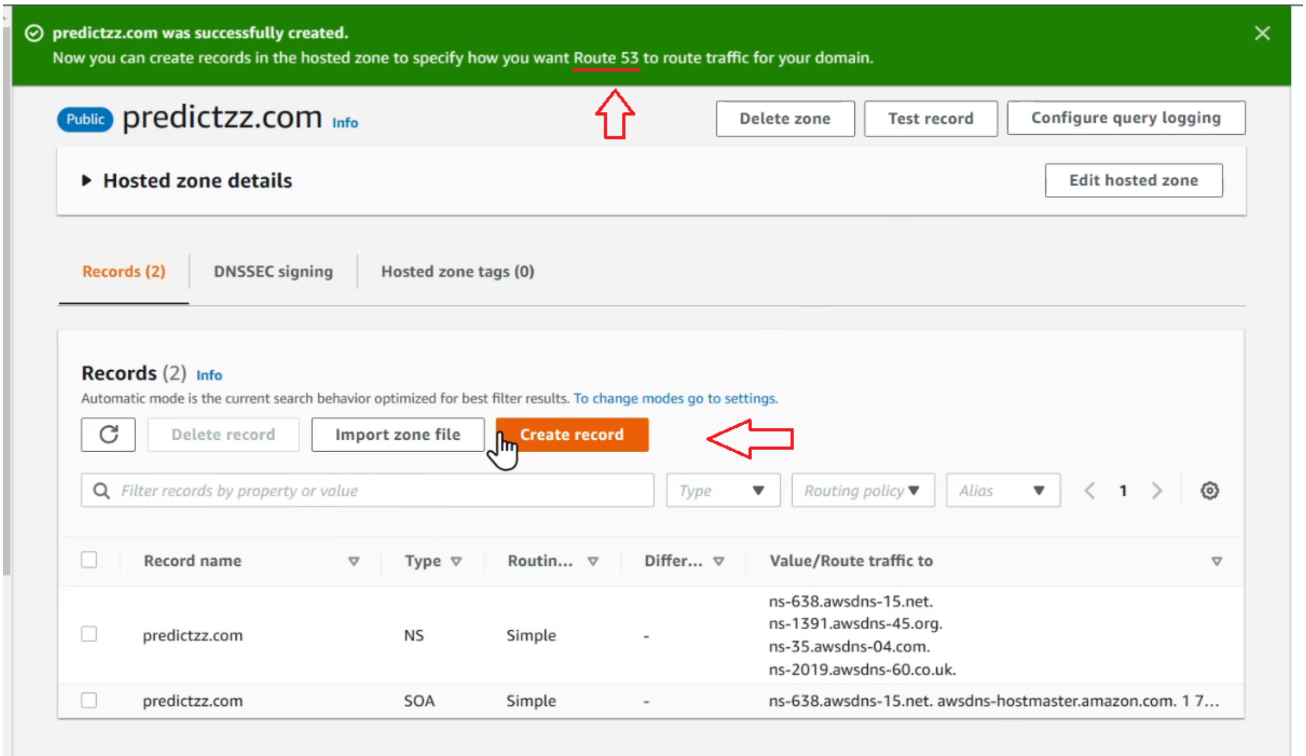


Figure 4.42 Record List

Figure 4.42 shows the record of the hosted zone. By default it have 2 record which is NS and Soa record. Now we need to create two A record for the hosted zone.

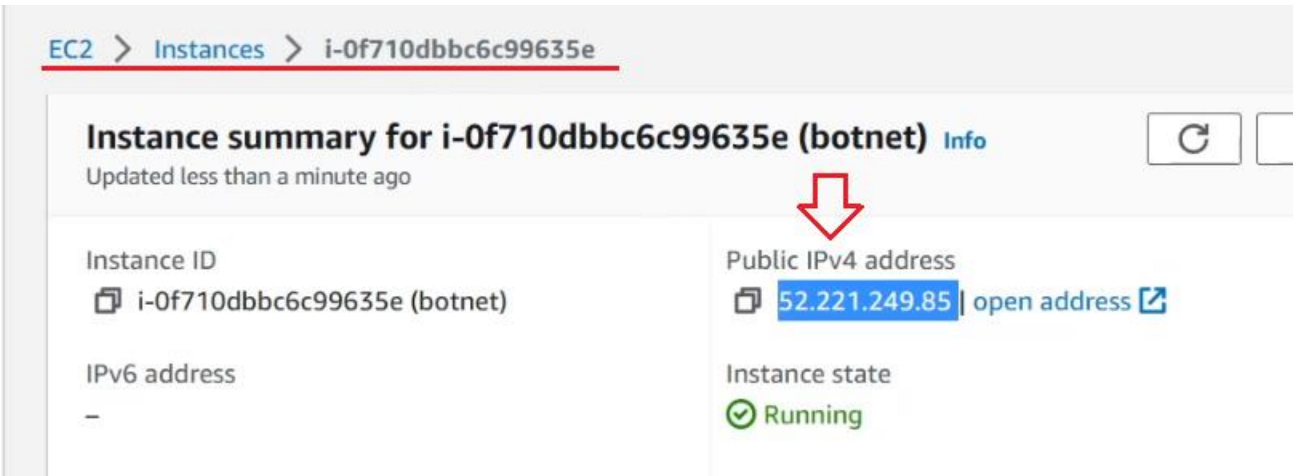


Figure 4.43 Public IP Address of Instance

Figure 4.43 shows the public ip address of the instance. This public ip address will be use to create a record.

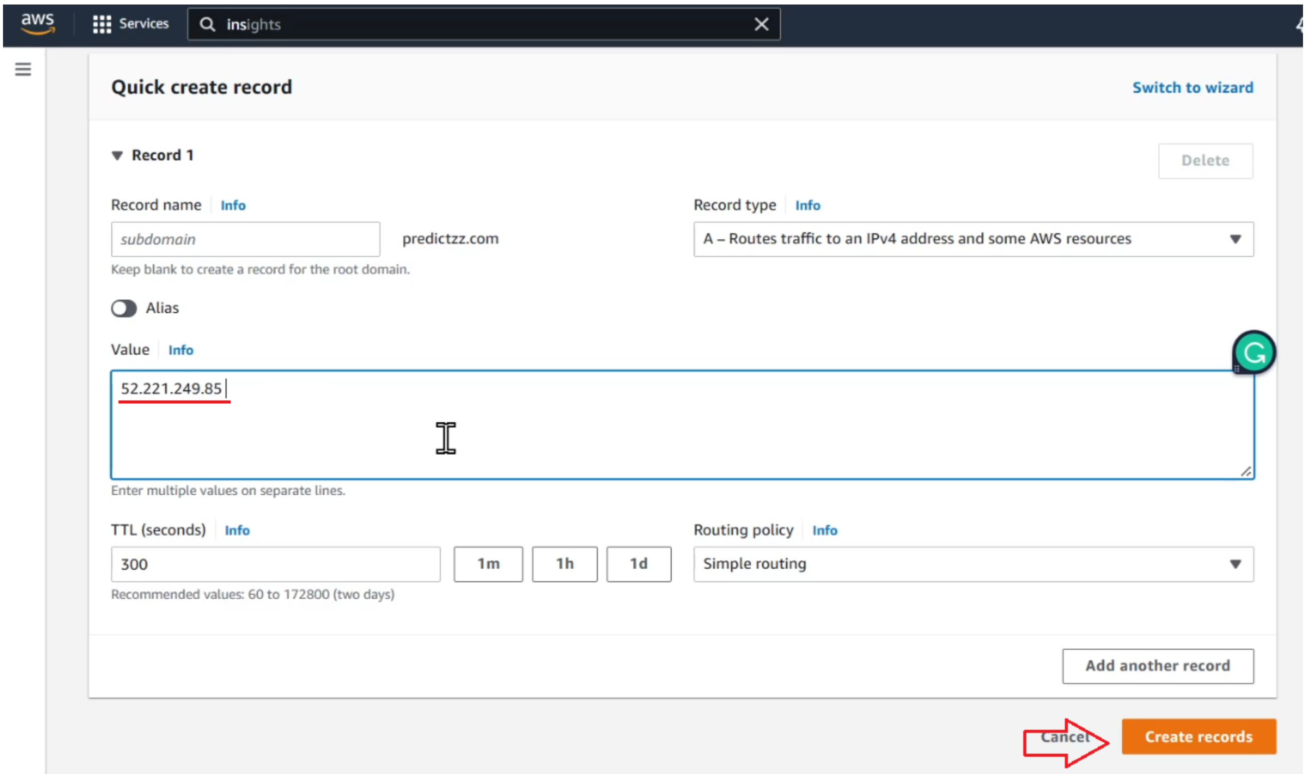


Figure 4.44 Assign Public IP Address

Figure 4.44 shows the public ip address of the instance that we get from Figure 4.43 will assign in the record. After assigned, click create records.

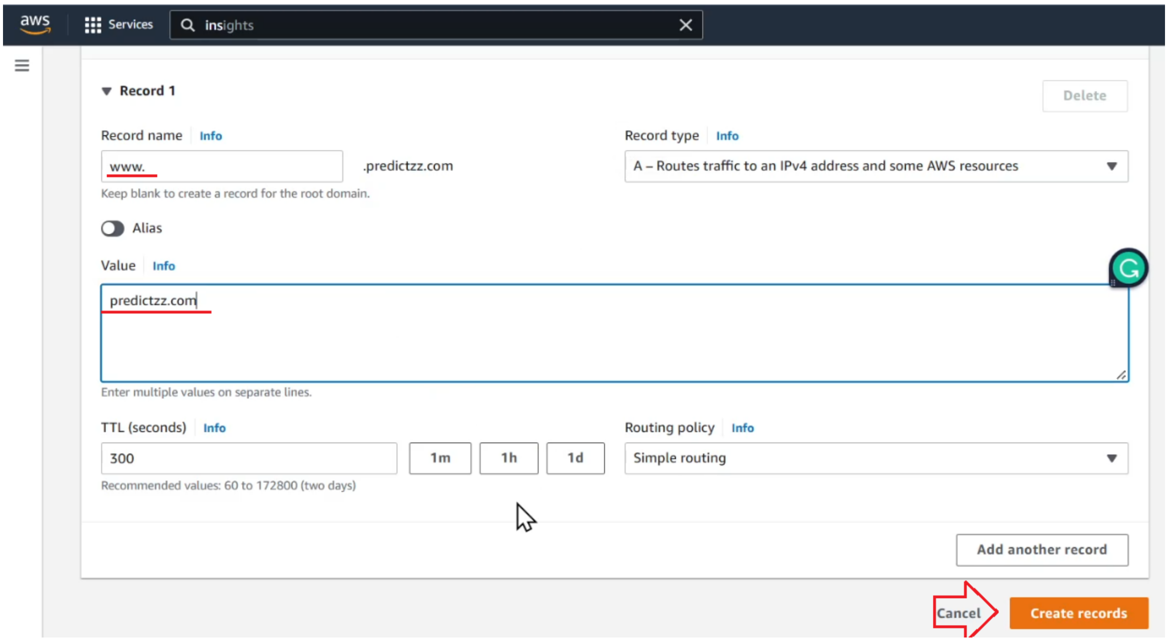


Figure 4.45 Assign Sub Domain in Record

In Figure 4.45 shows we assign a sub domain which is www for the predictzz.com domain in another record.

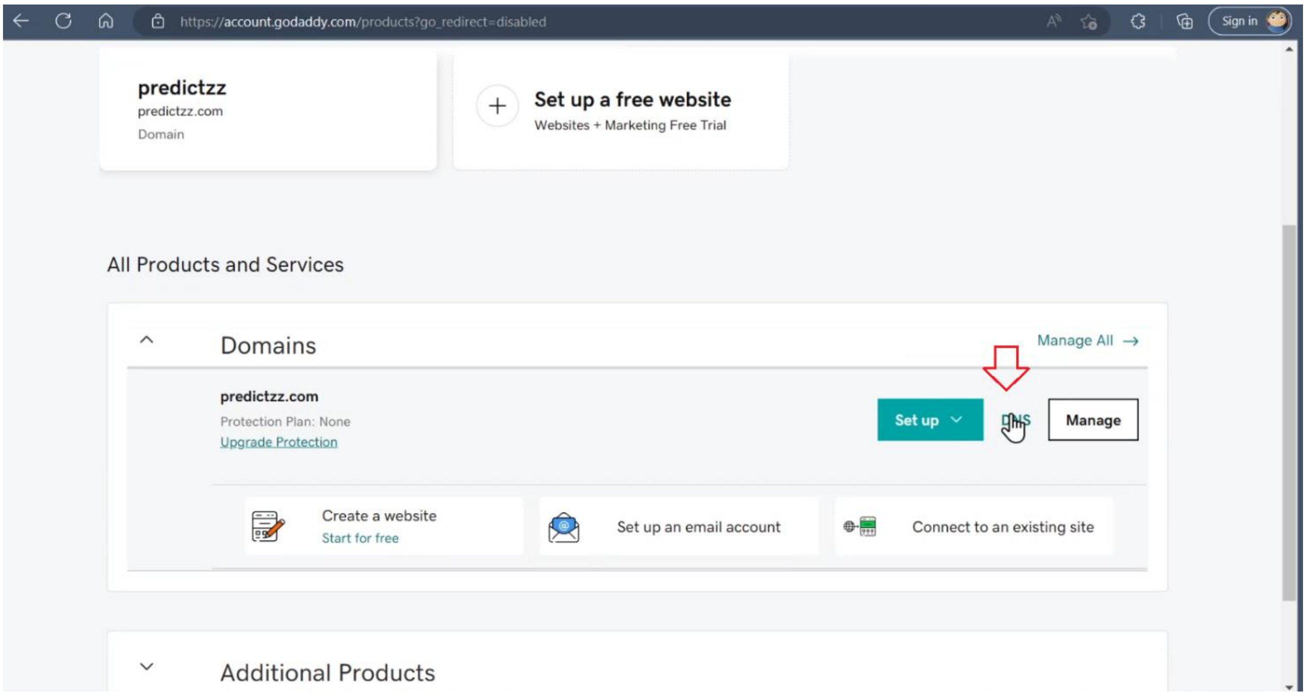


Figure 4.46 GoDaddy My Product Page

Figure 4.46 shows the My Product Page. We need go for the GoDaddy DNS management page to do nameserver setting by click DNS as shown in the figure.

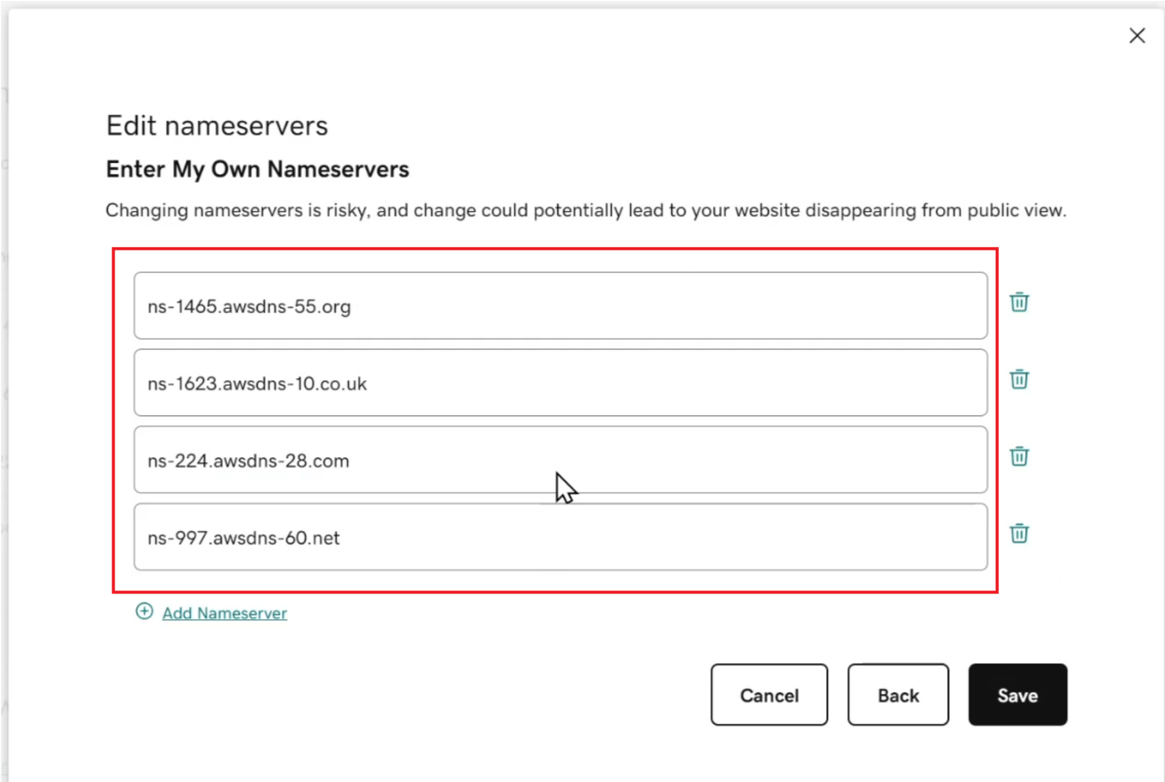


Figure 4.47 Edit Nameservers

In Figure 4.47, we need to update the name server from GoDaddy nameserver to AWS nameserver. The AWS nameserver is listed in Figure 4.48. After update the nameserver it will takes up to 48 hours to make the changes.

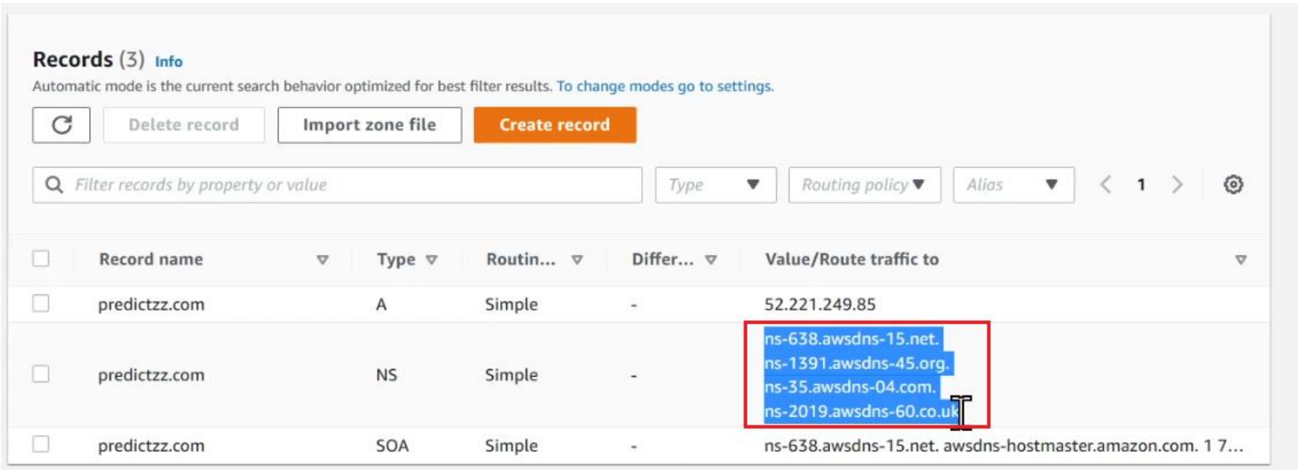


Figure 4.48 AWS Naemeserver

Figure 4.48 shows the AWS nameserver.

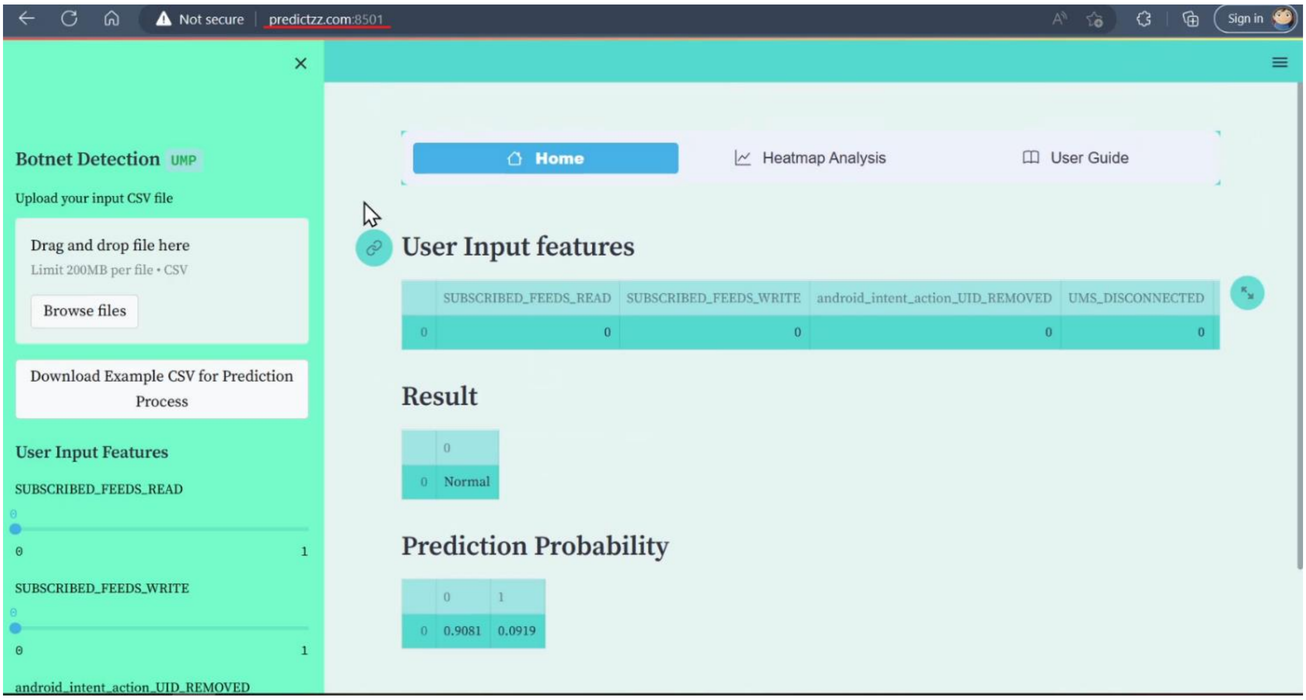


Figure 4.49 Botnet Detection Website

Figure 4.49 shows the web application successfully access using the predictzz.com domain name.

CHAPTER 5
CONCLUSION

5.1 Objective Revisited

1) To study feature selection of Product Moment Correlation Coefficient (PMCC) algorithm with heatmap for machine learning model classification and development.

We were able to accomplish this by putting forward the algorithm of correlative matrices, which was used to choose and detect the best characteristics in the heatmap. The viability of this approach was demonstrated in chapter 4 as part of the implementation and development phase.

2) To develop a Botnet detection system with Product Moment Correlation Coefficient (PMCC) with heatmap intelligent.

In order to develop the botnet detection system, we need to choose the most suitable feature. We got a different feature for each dataset we used, because each feature only related to the others if the attributes or characteristics had strong and important data. From Table 5.1 to Table 5.3, you can see the different things we can capture with PMCC.

Table 5.1 Features Selection for 1st Approach

Dataset	Botnets
1st Approach	14 features
Value	0.99
Best Features	1. SUBSCRIBED_FEEDS_READ 2. SUBSCRIBED_FEEDS_WRITE 3. android.intent.action.UID_REMOVED 4. UMS_DISCONNECTED 5. INPUT_METHOD_CHANGED 6. UID_REMOVED 7. ServiceConnection 8. bindService 9. Ljava.util.Timer.*schedule 10. Ljava.util.TimerTask 11. Ljava.util.Timer 12. AssetManager 13. Landroid.content.res.AssetManager

	14. getAssets

Table 5.2 Features Selection for 2nd Approach

Dataset	Botnets
2nd Approach	36 features
Value	0.90-0.99
Best Features	1. READ_CALL_LOG 2. READ_USER_DICTIONARY 3. WRITE_CALL_LOG 4. WRITE_USER_DICTIONARY 5. android.intent.action.PACKAGE_REPLACED 6. android.intent.action.BATTERY_LOW 7. android.intent.action.UID_REMOVED 8. android.intent.action.BATTERY_OKAY 9. android.intent.action.SCREEN_OFF 10. android.intent.action.SCREEN_ON 11. PACKAGE_REPLACED 12. UMS_CONNECTED 13. UMS_DISCONNECTED 14. BATTERY_LOW 15. .apk 16. .so 17. onBind 18. IBinder 19. Ljavax\/crypto\/Cipher 20. ProcessBuilder 21. Process. *start 22. Ljava.util.Timer.*schedule 23. Ljava.util.TimerTask 24. ZipInputStream.*close(25. ZipInputStream.*getNextEntry(26. SUBSCRIBED_FEEDS_READ 27. SUBSCRIBED_FEEDS_WRITE 28. android.intent.action.UID_REMOVED 29. UMS_DISCONNECTED 30. INPUT_METHOD_CHANGED 31. ServiceConnection 32. bindService 33. Ljava.util.Timer 34. AssetManager 35. Landroid.content.res.AssetManager 36. getAssets

Table 5.3 Features Selection for 3rd Approach

Dataset	Botnets
3rd Approach	47 features
Value	0.99
Best Features	1. READ_CALENDAR 2. SET_ALWAYS_FINISH 3. SET_DEBUG_APP 4. SIGNAL_PERSISTENT_PROCESSES 5. WRITE_CALENDAR 6. WRITE_SYNC_SETTINGS 7. PICK_WIFI_WORK 8. BATTERY_OKAY 9. HttpPost.*init 10. HttpUriRequest 11. createSubprocess 12. READ_CALL_LOG 13. READ_USER_DICTIONARY 14. WRITE_CALL_LOG 15. WRITE_USER_DICTIONARY 16. android.intent.action.PACKAGE_REPLACED 17. android.intent.action.BATTERY_LOW 18. android.intent.action.UID_REMOVED 19. android.intent.action.BATTERY_OKAY 20. android.intent.action.SCREEN_OFF 21. android.intent.action.SCREEN_ON 22. PACKAGE_REPLACED 23. UMS_CONNECTED 24. UMS_DISCONNECTED 25. BATTERY_LOW 26. .apk 27. .so 28. onBind 29. IBinder 30. Ljavax\/crypto\/Cipher 31. ProcessBuilder 32. Process. *start 33. Ljava.util.Timer.*schedule 34. Ljava.util.TimerTask 35. ZipInputStream.*close(36. ZipInputStream.*getNextEntry(37. SUBSCRIBED_FEEDS_READ 38. SUBSCRIBED_FEEDS_WRITE 39. android.intent.action.UID_REMOVED

	40. UMS_DISCONNECTED 41. INPUT_METHOD_CHANGED 42. ServiceConnection 43. bindService 44. Ljava.util.Timer 45. AssetManager 46. Landroid.content.res.AssetManager 47. getAssets

3) To evaluate the detection performance of the Botnet detection system.

This objective has been demonstrated in Chapter 4 under the results. The detection findings were obtained after all the methods in this paper was followed and implemented. Using the Coarse Tree Classifier technique, we may evaluate our final objective. The accuracy for the Botnet dataset with the different approach was 99.8, 99.6, and 99.6 respectively. Following that, the TPR after training is 0.998, 1, 1 and the FPR is 0.003, 0.041, and 0.041 respectively.

Table 5.4 Result of Coarse Tree Classifier

Classifier Algorithm	Number Features	ACC	TPR	FPR	AUC
Coarse Tree	14f	99.8	0.998	0.003	0.9983
	36f	99.6	1	0.041	0.9752
	47f	99.6	1	0.041	0.976

5.2 Limitation

Because the virus has the potential to spread and persist in the future, the list of Botnets may change. Furthermore, botnet families will grow and evolve, creating a new and more dangerous threat. As a result, more harmful botnets may attack the environment and network at a much faster rate. Even if we utilise feature selection as our model-building technique, knowing more about the dataset can help us make better and more accurate predictions. As a result, the dataset's characteristics must be enhanced in order to capture more necessary traits and traits from botnets in order to prevent them from creating a new family and to recognise the red flag in a faster and more accurate manner.

5.3 Future Work

Future studies can be incorporating various feature selection algorithms or methods to select the best features, as the features will be increasing and be spread. So that, machine learning (ML) needs to become more efficient and reliable, as many features are filtered out and the scan is based on features. Next, machine learning (ML) must retrain its classifiers to ensure that the spreading can be stopped and that it does not become more broadly disseminated faster.

REFERENCES

[1] “Botnet - Definition.” https://www.trendmicro.com/vinfo/us/security/definition/botnet (accessed May 30, 2022).

[2] S. Y. Yerima, M. K. Alzaylaee, A. Shajan, and A. Rosa Cavalli, “electronics Article,” 2021, doi: 10.3390/electronics.

[3] S. Y. Yerima and A. Bashar, “A Novel Android Botnet Detection System Using Image-Based and Manifest File Features,” *Electronics (Switzerland)*, vol. 11, no. 3, Feb. 2022, doi: 10.3390/electronics11030486.

[4] G. Wang *et al.*, “Feature selection for malicious android applications using Symmetrical Uncert Attribute Eval method,” *IOP Conference Series: Materials Science and Engineering*, vol. 884, no. 1, p. 012060, Jul. 2020, doi: 10.1088/1757-899X/884/1/012060.

[5] Z. Fang, J. Wang, J. Geng, and X. Kan, “Feature Selection for Malware Detection Based on Reinforcement Learning,” *IEEE Access*, vol. 7, pp. 176177–176187, 2019, doi: 10.1109/ACCESS.2019.2957429.

[6] N. H. Mazlan, N. H. Mazlan, and I. R. A. Hamid, “Evaluation of Feature Selection Algorithm for Android Malware Detection,” *International Journal of Engineering & Technology*, vol. 7, no. 4.31, pp. 311–315, Dec. 2018, doi: 10.14419/ijet.v7i4.31.23387.

[7] “Android Botnets: What URLs are Telling Us | springerprofessional.de.” https://www.springerprofessional.de/en/android-botnets-what-urls-are-telling-us/6871696 (accessed May 30, 2022).

[8] W. Hijawi, J. Alqatawna, and H. Faris, “Toward a detection framework for android botnet,” *Proceedings - 2017 International Conference on New Trends in Computing Sciences, ICTCS 2017*, vol. 2018-January, pp. 197–202, Jul. 2017, doi: 10.1109/ICTCS.2017.48.

[9] S. Hojjatinia, S. Hamzenejadi, and H. Mohseni, “Android botnet detection using convolutional neural networks,” *2020 28th Iranian Conference on Electrical Engineering, ICEE 2020*, Aug. 2020, doi: 10.1109/ICEE50131.2020.9260674.

[10] C. Tansettanakorn, S. Thongprasit, S. Thamkongka, and V. Visoottiviseth, “ABIS: A prototype of Android Botnet Identification System,” *Proceedings of the 2016 5th ICT International Student Project Conference, ICT-ISPC 2016*, pp. 1–5, Jul. 2016, doi: 10.1109/ICT-ISPC.2016.7519221.

[11] M. Yusof, M. M. Saudi, and F. Ridzuan, “A new mobile botnet classification based on permission and API calls,” *Proceedings - 2017 7th International Conference on Emerging Security Technologies, EST 2017*, pp. 122–127, Oct. 2017, doi: 10.1109/EST.2017.8090410.

[12] B. Alothman and P. Rattadilok, “Android botnet detection: An integrated source code mining approach,” *2017 12th International Conference for Internet Technology and Secured Transactions, ICITST 2017*, pp. 111–115, May 2018, doi: 10.23919/ICITST.2017.8356358.

Appendix

Gantt Chart