Final Report

CM3203 - One Semester Individual Project - 40 Credits

Creating a machine learning model based on network activity to detect attacks from a malicious webserver

Author - Balqees Ali Al-Ajmi

Supervisor - Amir Javed

Moderator - Martin J Chorley

A Final Year Project Submitted for the Degree of Bachelor of Science

Department of Computer Science with Security and Forensics Cardiff University

2021

ABSTRACT

Drive-by-download refers to attacks that automatically download malware to the user's device, usually without the victim’s knowledge or his consent. This kind of attack is carried out by leveraging vulnerabilities to web browsers and plugins. The damage could include leakage of data leading resulting in financial loss. Traditional antivirus and intrusion detection systems are low efficient against such attacks. The emphasis of machine learning techniques in today's computer technology is on the credibility of machine learning in the arts and sciences of malware detection. Since machine learning methods are efficient and simple to use, they are used to minimize malware activities. This project aims mainly at classifying malicious network traffic into either malicious or benign traffic from the given dataset and to evaluate the performance of different machines learning classification models, with evaluation metric to determine the best learning techniques. This model is trained with a certain dataset, which relies on the most common machine learning classification models which have been applied successfully toward the detection of network attacks can extract similarities and patterns in the network traffic such as Random Forest, Support Vector Machines in three different variations and Naïve Bayes. Feature selection was carried out during the project, and Python programming language was utilised for the entire project.

ACKNOWLEDGEMENTS

First, I want to say Alhamdulillah, I would like to take this opportunity to thank Allah for keeping in a good health that was necessary for completing this project during an unexpected time and for his guidance throughout my life and also to express my gratitude to the people who have supported me throughout the whole project. A massive thank you to my supervisor Amir Javed for his continued enthusiasm and invaluable support and guidance throughout the whole project and I do not doubt that without his support, the project would have been far less of a success. Another thank you to my family, especially my mother, who inspire me with confidence and beyond and I could not have done this *project and complete my bachelor's degree without their support, which I* will forever cherish. I would also like to thank all of my friends for the amazing memories we shared for making me feel home away from my home. I want to thank everyone, we have been through difficult times and we overcome the hard circumstances together.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Preface

Nowadays, The Internet and computer networks have become a significant aspect of organizations and everyday life. The rapid growth of Internet interconnections has caused a substantial rise in cyber-attack incidents that are often catastrophic and serious. Increasingly malicious activities are prevalent with the growth of dependence on computers and communication networks. In today's communication environments, network attacks are an important issue. Drive-by downloads are a typical method utilised by the abusers to silently install malware on a victim's machine. Once a target website has been armed with some kind of vulnerability (usually exploits by a browser or plugin, secret iframes and JavaScript) the attacker may lure or wait until his target is visited on the web page. In general, the compromised page looks very natural for the end-user, and the exploit runs and installs malware in the silent background on the victim's computer. The attacker will take action on his targets until the malware enters the target device.

Everyday Internet users are a target by a large number of attackers who are continually looking for vulnerabilities to do attacks on various sites for different reasons and purposes [14]. According to the research conducted by several researchers in 2010 states that the drive-by download attacks are considered to be one of the most significant types of attacks in which the attacker uses legal and illegal websites to spread malicious code [1]. A file is downloaded to the user device without trigger by exploiting the vulnerability of a web browser. The file usually includes a malicious code running on the target machine. This malware can be used to access sensitive information and steal it, build a backdoor or serve some possible malicious function. Leit and Cova assume that drive-by downloads are involved in the spread of most current malware infections. The ultimate purpose of a drive-by-download attack is to monitor the system of the client through exploiting Web browsers' vulnerabilities or its extensions forcing it to carry out undesirable operations [21]. Takata, Akiyama, Yagi, Hariu, and Goto believe that attacks

could result in data or financial loss is the particular reason for the circumstance the attacker controls the victim's machine [39]. In addition, if it were counted as a country, then cybercrime, which is predicted to be the third-largest economy worldwide after the US and China, will be a projected $6 trillion global harm in 2021[47]. Cybersecurity Ventures anticipates that over the coming five years, global cybercrime costs will rise by 15 per cent per year to reach USD 10.5 trillion annually by 2025, up from USD 3 trillion in 2015. This is the largest transfer of economic wealth in history, threatens creativity and investment opportunity is exponentially greater than the harm caused by natural disasters in a year and would be more lucrative than the global trafficking of all major illicit drugs combined [ibid]. Cybercrimes costs data damage and destruction, stolen money, loss of productivity, theft of intellectual property, theft of confidential and financial data, embezzlement, fraud, disturbances to the normal course of business following the attack, forensic investigations, restoration and removal of hacked data and systems as well as reputational harm.

Cyber-attacks are growing year on year; therefore, it is significant to avert attacks by rising the security of the network base and increasing protection of the data information, which could be achieved by using protected devices, such as, antispam devices, antivirus services, firewalls, IPS (Intrusion Prevention System), IDS (Intrusion Detection System). Anti-malware is software that have the ability to protects computer networks from malware by recognizing malicious networks that attempt to interact with the computers in this way to avoid any intrusion or harm by this malware. It significant to monitor and analyse network traffic to detect malicious activities and attacks to ensure stable and reliable functionality of the networks and provide the protection of user information. Therefore, Machine Learning algorithms have become widely popular in the detection of anomalous network traffic with higher levels of accuracy and adaptability in a wide range of situations and environments, with its advancement and adoption in many domains, including computer science [31]. Developing and applying machine learning models for malware detection would add a high level of security, the particular reason for that it can keep pace with malware evolution. Machine Learning-based anti-malware tools help to detect modern malware attacks and develop better scanning engines [13] [11].

1.2 Project Aims and Scope

The main purpose of this project is to create a machine learning model based on Network Activity to detect Attacks from malicious webserver from a dataset and to investigate the performances of different Machine Learning classification algorithms with evaluation metric to identify the best machine learning techniques.

The project scope focuses on implementing and examine the performances of different Machine Learning Classification algorithms able to distinguish the Network traffic in training and testing set, whether network activity is malicious or benign. The novel approach used in this project is implementing three different types of Machine learning algorithms with three different variations of one type. They are Random Forest, Support Vector Machine in three different variations (Linear kernel, RBF kernel, Sigmoid kernel) and Naïve Bayes. All Machine Learning algorithms have been chosen based on their popularity and their success that used in detecting drive-by download infected web pages based on extracted features. The data set used is network traffic data captured in a realistic environment. Since python is a powerful general-purpose programming language and the most preferred language to build machine learning task, All the implementation of the project will use python as a programming language and will be implemented and tested on a Windows Operating System device. More information will be discussed later within the implementation section.

1.3 Intended Audience

The intended audience and beneficiaries from this project are the researchers, analysts and individuals who are interested in researching the field of security of applications built on Machine Learning algorithms and more specifically in the field of malware detection on network traffic.

1.4 Report structure

Chapter 1 introduces the project, its aims and objectives and the audience who may find it interesting. **Chapter 2** provides a short background of Machine Learning and various Machine Learning approach utilised for malware detection and measures for evaluating Machine Learning classifiers. Machine Learning basics and approaches, evaluation measures of Machine learning models as classifiers and related work. **Chapter 3** discusses

the requirements and design of the provided solution. **Chapter 4** explains the implementation of various Machine Learning algorithms on using the specific dataset down to the coding level. **Chapter 5** presents the test cases that were undertaken to test the implemented solution. **Chapter 6** discusses the results of the Machine Learning algorithms utilised to solve the problem along with the Limitations of the approach taken. **Chapter 7** discuss potential future work that could be undertaken to improve the project. **Chapter 8** concludes the main findings of the project. **Chapter 9** discuss the reflection on Learning gained from completing this project.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

2.1 Introduction to Machine Learning

Machine learning is a separate field of computer, which utilised stat structural patterns recognition and artificial intelligence methods in any task that requires gathering or extraction of information from a massive source of data that surround us. The term Machine Learning means the acquisition of the areas of structural patterns recognition from examples for future prediction or making decisions, explanation and understanding purposes using data pattern recognition and artificial intelligence methods [26]. Arthur Lee Samuel is the first American pioneer in the field of computer gaming and artificial intelligence popularized the term "machine learning in 1959 as: "the field of study that enables computers to learn without specifically being programmed" (Samuel 1959). In 1997, the scientist T. Mitchell described Machine Learning in a more formal definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."[40][3].

Over the last years, Machine Learning approaches are substantially becoming a high demand in many different industries and businesses for the exact purpose of obtaining meaningful data insights and automation analysis [5], With technology today, Machine learning is the process that powers a wide range of services that touch our daily lives, as machine learning technologies extensively used by most industries in multiple domains; As an example, financial services, health care, transportation, gas and oil, government, and retails [ibid]. The general idea of any machine learning task is to train the model, based on some algorithm, to execute a certain task: Classification, Clusterization, Regression, etc. In terms of classification, Machine Learning approaches to supply an automated, adaptive approach that extracts information from provided training datasets using a

Machine Learning model such as Classification, to utilise the information gained in the categorisation of the unseen data, as illustrated in Figure 2.1.

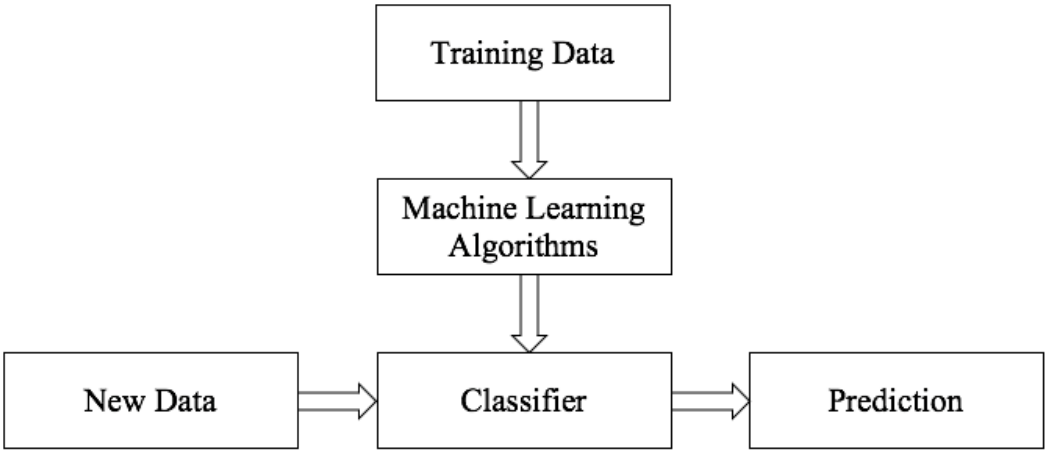


Figure 2.1 Machine Learning Flow

Where Classification is a method of data analysis that extracts models, which categorize major classes of data. These models are known as Classifiers. A classifier in machine learning is an algorithm that learns from a given data and uses rules by machines to classify observations into one or more of a set of "classes." Classes are sometimes called targets/ labels or categories. It works by identifying specific features that aid in discriminating between observations. The classification algorithm process is to input data and divides it into two groups, the training dataset and the other the validation-test dataset [45]. It is important to split data into a training portion and a test portion, thus the model can be trained on the first and then tested with the testing data. Training is done based on the input dataset and the built-in model that is built is subsequently utilized to make predictions. The training data is comprised of multiple features. Then, the A classifier is to apply all or chosen some of the features of the training data to understand how given input variables relate to the classes; next, rules will be produced using the training data to classify observations. The Machine Learning classification process is illustrated in figure 2.2. The performance of this model depends on the initial task and the implementation. There are multi applications of Machine Learning classification techniques that exist and have been used by researchers in many fields, including fraud detection, medical diagnosis, credit approval, performance prediction and target marketing [2][ibid].

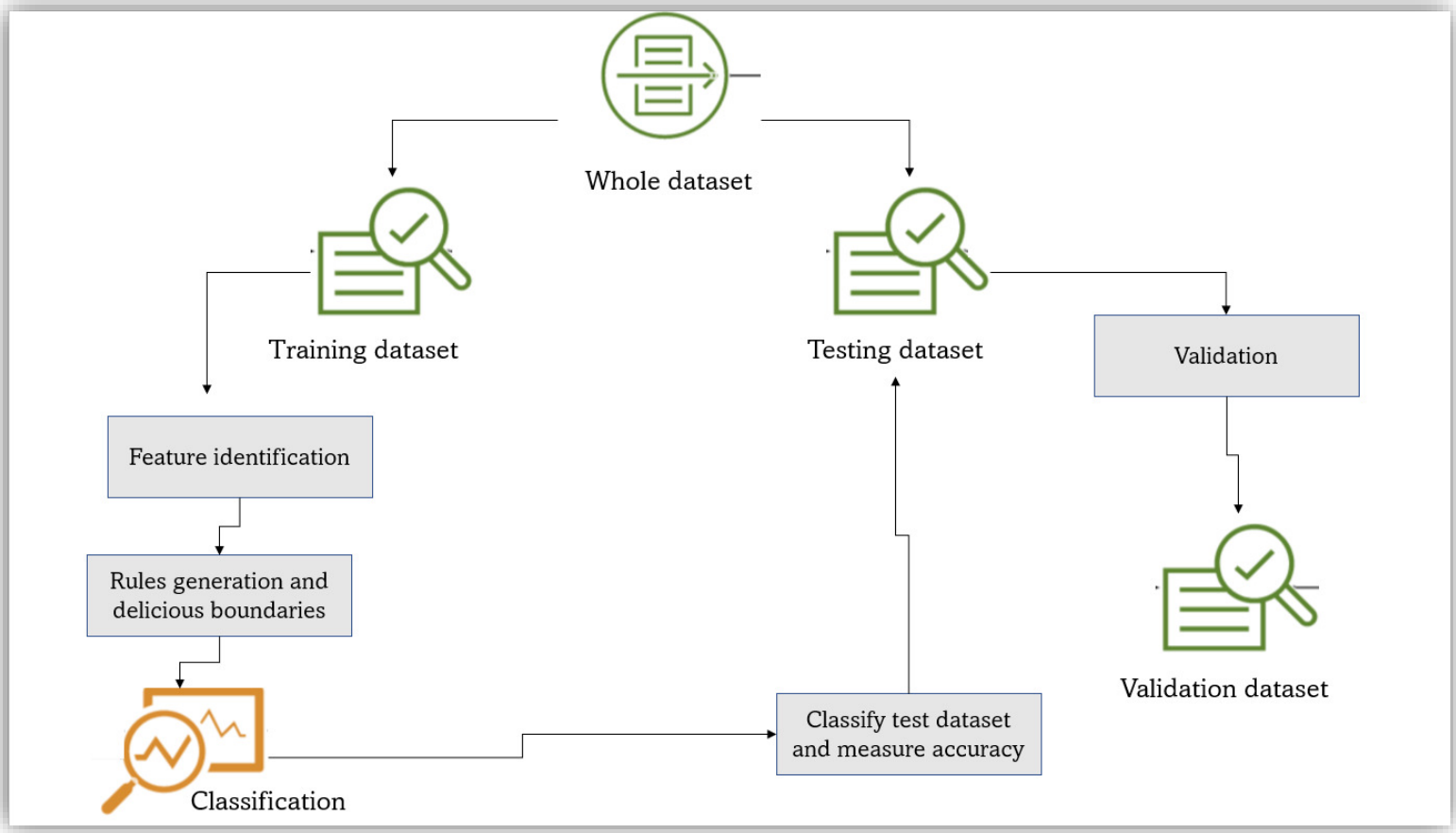


Figure 2.2 Machine Learning classification

There are several Machine Learning classifications techniques are exist, some of the most popular utilised Machine Learning classification model that is going to be used in this project and will be discussed and presented in the upcoming sections.

2.1.1 Machine Learning Techniques

Learning is, of course, a very wide range domain. Consequently, the domain of Machine Learning models has branched into several subfields dealing with various types of learning tasks, Supervised and Unsupervised Machine Learning which are mostly used [18]. These types of machine learning are applied based on the availability of the observations which are already labelled and classified [18]. These models are applied for various purposes, and each type of learning model is implemented to perform either descriptive or predictive analysis depending on the selected algorithm, type of analysis required to solve the problem with the taking into consideration the type of dataset used for the analysis. The Supervised Machine Learning models will be discussed in the following sections with their uses.

Supervised Machine Learning Model

Supervised machine learning is the construction of algorithms that can produce general patterns and based on training a data sample from the data source with correct classification already assigned [25]. as shown in Figure 2.3, Supervised Machine Learning is learning in presence of a training dataset that is fully labelled as benign and malicious for the learning and training task of the model. That would provide an answer key that the algorithm can use to test the accuracy of training data[34]. Therefore, Classification is a typical example of Supervised Machine Learning. Network traffic is an example of a labelled training dataset, which consists of files that are used with two groups of labels of "malware" or "benign" traffic for the learning and training task of the model. Classification could be applied to this type of labelled dataset for a supervised Machine Learning algorithm. One important step in building a classification model by applying machine learning methods. Our project is focusing on creating Supervised Machine Learning will be utilised by applying classification methods, to learn the extracted features and that associate strongly with a particular class for its detection.; therefore, it gains the needed knowledge to categorise a new dataset that is not labelled [38]. In this project, a supervised machine learning approach will be used for classification.

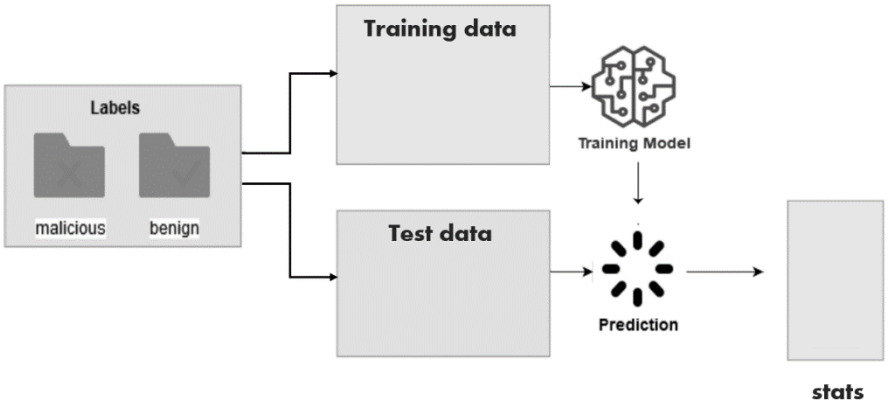


Figure 2.3: Working model of a supervised learning

2.2 Machine Learning Approach to malicious traffic detection

Malware detection is considered a binary classification task. In this project. 0 and 1where denote benign and malicious traffic, respectively. Therefore, Machine Learning techniques and statistical approaches are used to build classifiers that will be trained using labelled training data that contains network traffic captures packets and labels. The most commonly used classification algorithms of Machine Learning models for detection purposes are Random Forest (RF), Decision Trees (DT), Support Vector (SV). Naïve Bayes (NB), and Neural Networks (NN). As suggested by Hyo-Sik Ham and Mi-Jung Choi, some of the most popular Machine Learning classification algorithms that are utilised for detecting malicious traffic are Random Forest, Support Vector Machine, Naïve Bayes [16]. The following sections will briefly discuss each of the classifiers mentioned above.

2.2.1 Random Forest Classifier

Random Forest (RF) is a supervised machine learning algorithms [41]. It requires almost no data planning and modelling but normally gives accurate results. Random forest is an ensemble learning model for classification, where the model constructs a strong learner by employing a set of decision- trees classification as well as regression algorithms to evaluate a massive number of decision trees (known as a Forest) generated randomly from the data [8]. More specifically, this collection of decision trees produces better precision in prediction. In particular, it takes the predictions from these trees and then chooses the best employing a voting mechanism, as illustrated in Figure 2.4 [20]. The Random Forest algorithm also produces variable significance. The significance of each variable in the random forest can be measured in two ways. Firstly, by measuring how much the accuracy reduces when the variable excluded. Secondly, by calculating the decrease of Gini impurity when a variable chose to divide a node [ibid]. After training the Random Forest classifier, labels for a new dataset with the same set of features as the training data set can be predicted [ibid]. A well-calibrated model can have a high level of precision in generating predictions on the new dataset.

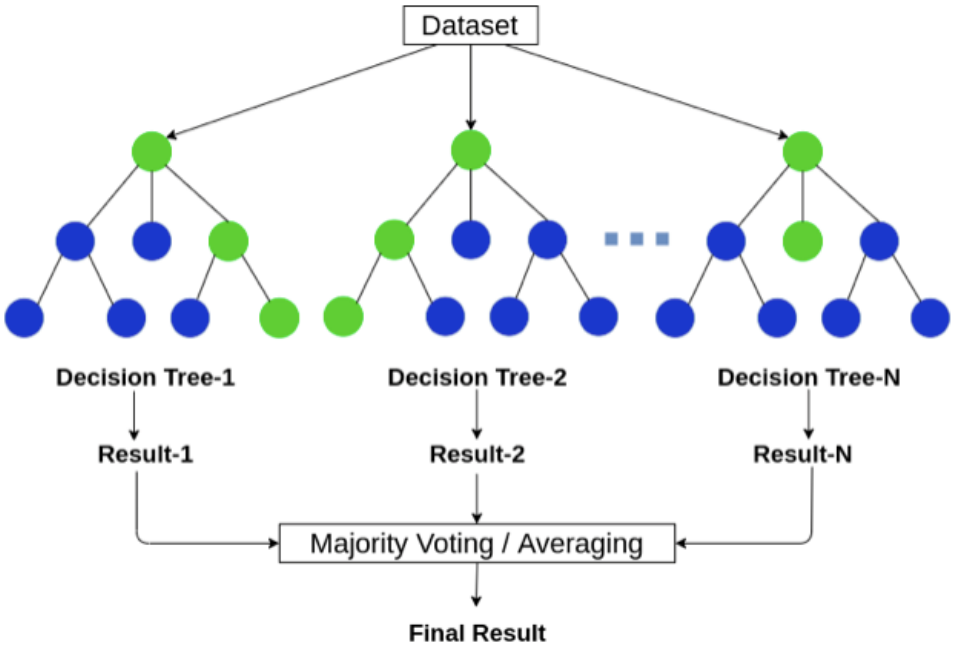


Figure 2.4 Voting mechanism in Random Forest algorithm

2.2.2 Support Vector Model Machine (SVM)

Vector Machines Support (SVM) is another strong machine learning algorithm that is generally utilised for classification problems in wide fields and yielding a great rate [23]. The main task of a Support Vector Machine is separating the data into different classes using an optimal hyperplane (boundary). Hyperplane can be linear or non-linear depending upon how the data is distributed in the dimensional space [22]. In a binary scenario, where the labelled dataset contains two classes; for example, benign and malicious, the algorithm generates a boundary to divides the two from each other (benign and malicious) by maximising the distance called margin between the separating hyperplane and the closest data points [17]. Data that are near to the hyperplane are called support functions, they represent the hard-to-classify cases and they have a direct bearing on the optimal location of the hyperplane. The optimum location of a hyperplane is determined by minimising the distance of the points from the hyperplane; this distance is called margin. A good margin for both classes is one where this separation is larger.

The reason behind choosing the hyperplane with the greatest margin is because it makes the classification correct for testing data that are close, but not identical to the training data [ibid]. A simple example is shown in Figure 2.5 illustrates the separating line

(hyperplane) and margin in simple terms, where the samples belong to the two-class malicious or benign traffic. There are different mathematical sets of Support Vector Machine algorithms that are given as the kernel. The kernel's function is to take the data as input and convert it into the required form. [43] several SVM uses different kernel functions; an instance of kernel utilised in this project are, Linear kernel, Radial Basis Function kernel (RBF), Sigmoid kernel [ibid].

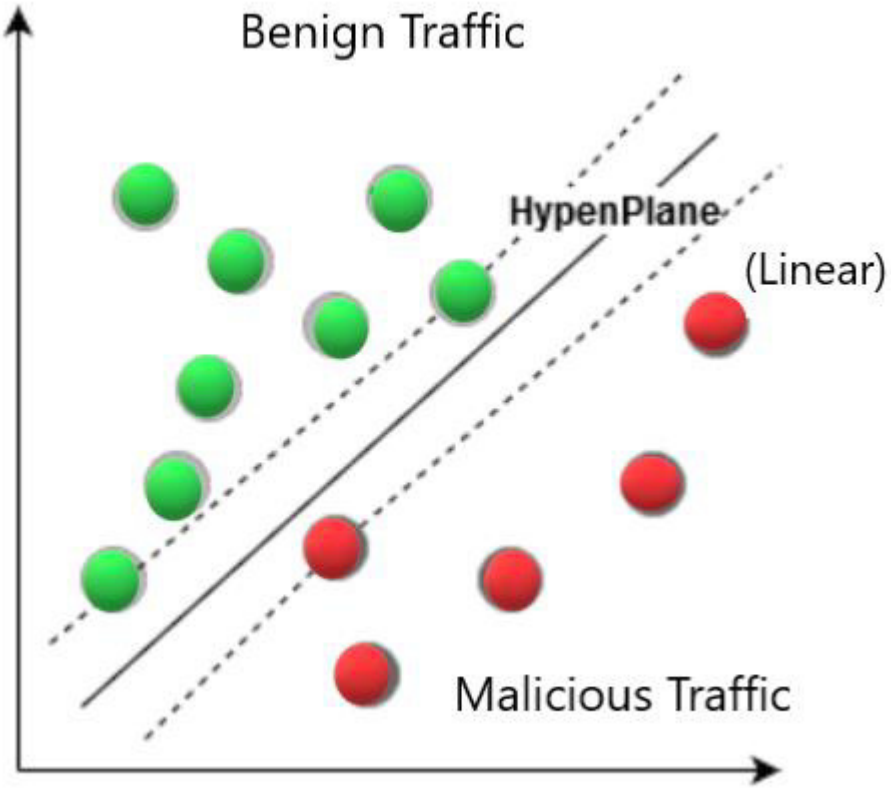


Figure 2.5 Support Vector Machine

2.2.3 Naive Bayes Classifier

The Naïve Bayes method is a supervised learning model that applies Bayes' theorem for predicting the conditional independence possibility of classes of various features for provided sample [32]. It is considered one of the popular prediction classification models [41]. The Naïve Bayes machine learning classification model is a probabilistic and assumptive approach, where it measures input training data with the assumption of conditionally independent of each other for a given class label [3]. Naive Bayes is a simple technique for given data, it will measure the prediction result based on the

maximum likelihood probability of features instances of training data sample [35]. Naive Bayes is a simple model and especially useful for a very large data set sample. Along with simplicity, Naive Bayes is recognized for its simplicity and superiority even to very advanced and highly sophisticated classification methods [30]. Bayes theorem provides a way of calculating the posterior probability P(c|x) from P(c), P(x), and P(x|c). Look at the equation below:

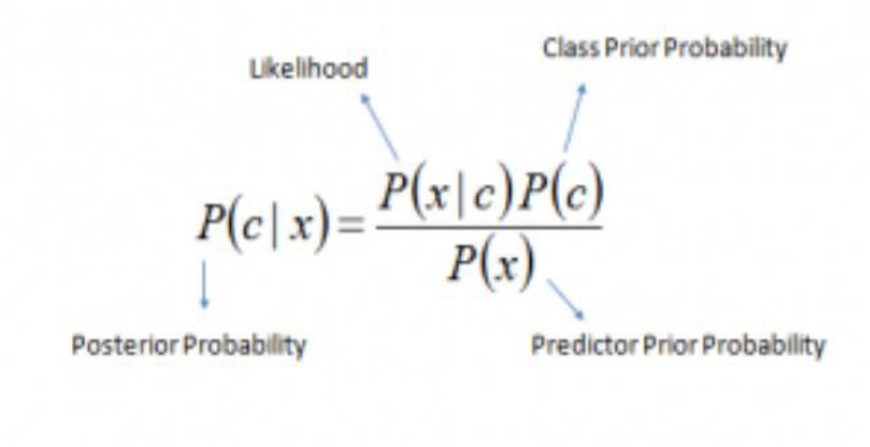


Figure 2.6 Bayes Theorem Equation from [35]

The P (c/x) represents the Naïve Bayes posterior probability and it includes main elements that are

P(c) , P(x) and P(x/c) [35].

- Both P(c) and P(x) represent the prior probabilities without regard to each other
  - P(c) is the previous likelihood of class label.
  - P(x) is the previous possibility that a given feature set appeared

- P(x/c) is the previous likelihood and predicator prior possibility.

In addition, Naïve Bayes declared that individual feature instances are not dependent as illustrated in Figure 2.6 [35].

2.3 Learning Evaluation (Evaluating what has been learne)

This section will include the evaluation technique for evaluating the performance of the machine learning model.

2.3.1 Confusion Matrix

Evaluation is the main key to success in Machine Learning [45]. To decide which classification algorithm is suitable for a particular classification problem or evaluate how well different models perform and compare one with another, A confusion matrix table is utilised, that indicates how successful is the classification model by summarising the prediction results. Confusion matrix, where outcomes are defined as follows,

• **TP**: correct classification of a positive class (The result of the model when it correctly predicts the positive class)

• **TN**: correct classification of a negative class (The result of the model when it correctly predicts the negative class)

• **FP**: incorrect classification of a positive class that is actually negative (The result of the model when it incorrectly predicts the positive class.)

• **FN**: incorrect classification of a negative class that is actually positive (The result of the model when it incorrectly predicts the negative class)

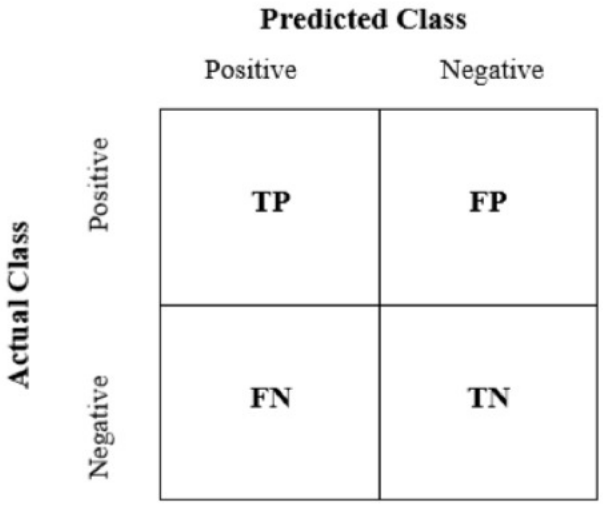


Figure 2.7 Confusion matrix for Binary classifier

The matrix contains two classifications against each other that are the predicted classification and the actual classification in form of four types of prediction outcomes as mentioned previously. Furthermore, These direct outputs of classification prediction are usually modelled in a Confusion matrix of two dimensions as seen in Figure 2.7. A good performance indicator from the Confusion matrix would be a higher number of prediction values for both TP and TN, this shows a good performing learning algorithm and relative to the total correctly classified for the testing set.

Generally, the Confusion matrix is useful as an accurate indicator for the model's outcome classification results. Moreover, other performing measures are utilised as performance comparison metrics such as Accuracy, Recall, Precision, F1-score and AUC to measure a classifier's performance. Each evaluation metric simply summarises the confusion matrix individually as seen in the following list that will be mentioned in the next point [45],

2.3.2 Performance Indicator

Accuracy: Accuracy is the overall success rate that is calculated by dividing the number of correct classifications (TP and FP) by the total number of the classifications, the best value is 1 and the worst is 0 [45].

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

Precision: specified as the number of correct Predicative value returned by the machine learning model. It is the ratio of correct positive predictions to the total number of positive examples in the dataset and it can be calculated by confusion metrics with the help of the following formula:

$$Precision = \frac{TP}{TP + FN}$$

Recall: represented as the number of classifiers to find all the positive instances. It is the ratio of correct positive predictions to the total number of positive examples in the dataset and it can be calculated by confusion metrics with the help of the following formula:

$$Recall = \frac{TP}{TP + FN}$$

F1 score, also known as F-measure. It represents the average of overall accuracy performance and it is calculated via precision and recall with equal weights. Therefore, this score takes into consideration both false positives and false negative. Intuitively, F1 with uneven class distribution is more useful than precise. To put it in other words, accuracy works better if false positives and false negatives cost the same. It is more accurate to consider both accuracy and reminder when they are different [10]. It can be measured by confusion metrics x with the help of the following formula:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

AUC stands for "Area under the ROC Curve.

ROC is a graph representing the performance of a classification model in all thresholds. This curve plots two parameters;

- True-Positive rate as a synonym for a recall
- False-Positive rate.

More area underneath the entire ROC curve, the better the classifier. Figure 2.8 illustrates the ROC curve and the AUC [8 B].

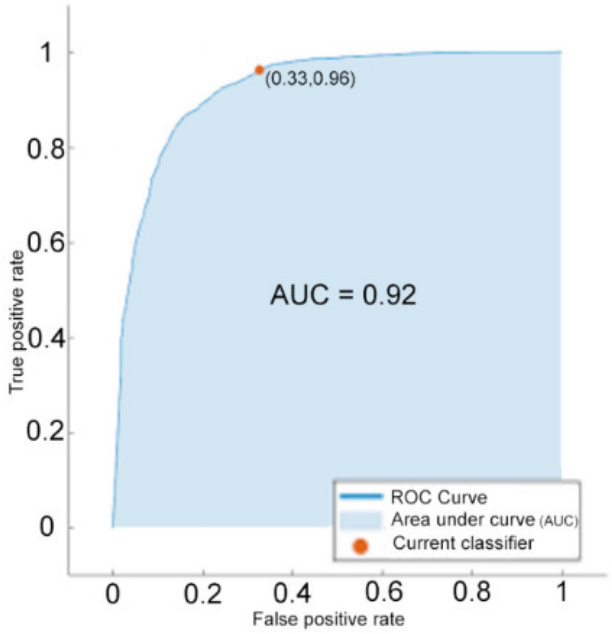


Figure 2.8 ROC curve and the AUC

2.4 Python ML libraires

This section will cover the tools that will be used for the development of the project. in this project for implementation purposes, research on supporting libraries for Machine Learning had to be taken. Python is considering as a high-level programming language that grants user to focus on the important function rather than programming tasks. The simple syntax rules make it easier to keep the code readable and maintainable. Python provides many open-source Machine Learning libraries and modules available that commonly supports the programming of Machine Learning models and algorithms. such as Scikit-learn library, SciPy, NumPy, Matplotlib, Pandas, Seaborn.

Scikit-learn library is considered the most popular library supporting the development of different machine learning models. Scikit-learn library consists of two basic libraries of Python, NumPy and SciPy, which provides a set of algorithms for common machine learning, including clustering, regression, and classification. Pandas is a Python Data Analysis Library utilised mainly for data manipulation and analysis. In particular, it provides data structures and operations for manipulating numerical tables and time series [36]. Panda will also be used to construct a panda data frame with a tabular data structure (rows and columns). NumPy is another essential package, which is considered as the core library for scientific computing in Python language that provides useful features and offers speedy computation and performance of complicated functions

working with arrays. NumPy library handles multi-dimensional data, where learning NumPy is the first step on any Python data scientist's journey. For the purposes of this project, Scikit-learn will be utilised, which is an actively used machine learning library for Python. There is a simple integration with various machine learning programming libraries like NumPy and Pandas to provide useful system [ibid]. And also, it includes essential and helpful techniques for machine learning such as, splitting data and measuring performance. Finally, Seaborn and Matplotlib both libraries used to visualise attractive graphs and to be applied for statistical purposes [44].

2.5 Existing solutions

There are several existing solutions of malware detection techniques that are relevant to the area of this project. however, increasing diversification of malware and its subtypes has made it significant to develop newer methods for better efficacy. Therefore, there is an urgent need for a generic set of features that are known to be helpful for malware detection and classification methods with a more comprehensive and up-to-date dataset. The following lists the most notable and recent proposed strategies by researchers in the area of characterization, track and analysis of drive-by download attacks. In the next paragraphs, each solution will be discussed and highlighting its advantages and disadvantages briefly. Kishore et al. (2014) created a browser extension to detect drive-by download attacks by observing webpages before they are loaded by users. If the system identifies the page as malicious, it will notify the user and take appropriate action based on his decision. This method depends on predefined malicious codes. As a result, it is identified as a signature-based classification system that can only detect well-known attacks. Unfortunately, attackers are constantly changing their techniques, therefore a signature-based detection system might not be the best approach [19]. In 2013, Le, Welch, Gao and Komisarczuk suggested a system that continually keeps tracking Internet Explorer 6 to detect the download popups by using window, mouse, and keyboard tracking modules. After the detection and tracking process, the system acts based on the file filter module. The main shortcoming of this system is that it does not consider downloads that occur in the background without the user's, knowledge, which is the case in most drive-by download attack scenarios. Thus, this system is built based on Internet Explorer 6, which is obsolete [42]. Priya et al. (2013) suggested a static method focused on elicitation from created databases of web contents with a parser written in MATLAB.

For less than 1000 manually collected URLs, a few classifiers were used to forecast the decision. They used K-Nearest Neighbour, Regression Tree, and Support Vector Machine classifiers. The features chosen are quite simple, such as the number of tags, the number of white spaces, and the placement of items within the text. During the testing process, they had an 87.22 per cent success rate with SVM, an 87.05 per cent success rate with Regression Tree, and an 87.74 per cent success rate with the KNN classifier. The limitations of this strategy are the small number of features and classifiers, as well as the failing to keep up with updates in malware attacks [29]. The studies were mentioned Previously, illustrate that there is a high demand for analysing network flow and there is not a single trivial solution for application classification or malware detection in network analysis. According to the success of previous research, it will be used to develop a machine learning model based on network activity to detect attacks from the malicious web server. This project is unique from other work, it has considered numerous approaches previously studied and examined by researchers. The approach of this experiment is to solve the problem by applying the most common malware detection classification model from a separate category of Machine Learning algorithms (Random Forest, Naive Bayes and Kernel à Support Vector Machine). In addition, three several types of the Support Vector Machine have been considered, which are, Linear kernel, RBF kernel, Sigmoid kernel. Another difference is that the developed solution will contain using several selected feature have been extracted from a **Pcap file** including classifiers performances evaluation with the combined result and comparing the performance of the provided classifiers to decide the best model produced. Additionally, One of the important Machine Learning algorithms is the Random Forest classification model, since different studies included Random Forest algorithms and it has suitability in multiple domains such as malware detection, and it also prevents overfitting.

2.6 Research Question

Aims

The main purpose of this project is to create a machine learning model based on Network Activity to detect Attacks from malicious web server from a given dataset and to investigate the performances of various Machine Learning classification algorithms with evaluation metric to identify the best machine learning techniques.

Research questions

To demonstrate the achievement of the stated aims, this project will explore common techniques employed to detect malicious traffic by implementing the most adequate Machine Learning classification models and applying appropriate evaluation metrics to evaluate the classifiers and to determine the best model which successfully detects an ongoing Drive-by download attack based on network activity.

CHAPTER 3

SPECIFICATION AND DESIGN

In order, for the project to be successful, sufficient time must be spent on implementing the solution. Therefore, This chapter of the report presents a clear explanation of the design of the implemented solution to be developed. In addition, It discusses the software requirements specification and initial plan of the design. Moreover, it mentions the system design and the methodology of this project.

3.1 Software Requirement Specification

The Software Requirement Specification for this project includes several functional and non-functional requirements concerning the technical implementation of the software design. These requirements are useful to decrease the cost and time of the project as they allow the evaluation of the success of the final result.

3.1.1 Functional Requirement

Data Type: The System must take network traffic data from the Comma Separated Value (CSV) file as an input with their labels and classify them into malicious or benign. This CSV file includes features is extracted from **a Pcap file**.

- **Pcap file** which is obtained from using Wireshark software is a capture of traffic network based on drive-by-download, this required for the system for training and testing the classifiers (more details of the **Pcap file** will be explained in chapter 4).

Data flows through the system: The system shall take data from the user input as CSV files of raw network traffic and their labels for training and testing classifiers.

- Raw network traffic files will be needed for training and testing the classifiers. Therefore, the system will get the data from the user input by specifying the name of the file that
- The system shall pre-process the given raw network traffic data.

Data flows through the system: The system must pre-process the given raw network traffic.

The system will pre-process the given raw network traffic:

- Once the raw network traffic that contains categorical (object), numeric (int64), and float64 type of values are provided by the user, the system will pre-process them. a pre-processing step will take place to output clean data with only numeric values to prepare the data for the Machine learning classifiers models. The next step will be feature selection, which will involve choosing appropriate features for each classifier model.

Algorithms: The system will provide three different classification algorithms with three different variations of one of the provided classifiers.

- The system will have Five different classification algorithms, Random Forest, Naïve Bayes, SVM Linear Kernel, SVM RBF Kernel and SVM Sigmoid Kernel.
- Train, test and evaluate each model and select the best performing Machine Learning Model.

Evaluation: The system must allow evaluating trained classifiers to find the best performing model (performed as a classifier).

- The system will display the results of calculating the Classifier's Accuracy, Precision, Recall, F1 score, and AUC for each training and test set.

3.1.2 Non-Functional Requirements

The non-functional requirements are defined in the following list,

• ***Reliability:***

There will be no errors in the system, and it will be available to use all the time. There will be performance indicators will be provided such as; Accuracy, precision, recall, F1 score, AUC, and confusion matrices, which are also indicators of reliability.

• ***Usability:***

The system will execute three models to be used for classification, it will be easy to use and adjust to.

The introduced models will be trained with pre-labelled data.

These models will be evaluated using a test set from the data.

• ***Speed:***

Pre-processing the datasets, creating the classifiers, testing and validating and evaluating the classifiers should be reasonably quick.

• ***Size:***

The system will not be larger than 1000 megabytes and will have a small impact on computer memory.

• ***Re*-usability:**

The implementation of the project is broken down into multiple modules that can be reusable in other projects or systems.

All models are taking column features from a panda data frame, and most type of files like CSV files can be saved as panda data. Therefore, these models can be used in different project with a different type of files.

3.2 System Implementation Architecture

The produced system will be implemented based on the given data set was captured in a cyber range environment. The data set was called contains 6077 Malicious network traffics out of 11808 network traffics. The system will primarily classify these networks traffic depends on its labelled, 0 for benign and 1 for malicious traffic. The implementation of the project will be done and achieved using Python Programming Language and the script will be written in one Jupyter notebook file called FinalMLClassificationModels.**ipynb.** This file will consist of sections that work sequentially to feed into the next or acts as a standalone module to be reused elsewhere in the program. There are five sections within FinalMLClassificationModels.**ipynb** file and they are as follow:

DataReader: loading csv file: read the file.

Exploratory Data Analysis (EDA): which is an approach to analyse the provided data set and determine their main attributes.

pre-process data which prepare data for the machine learning classifiers models.

machine learning models (classifiers) contains three main models Random Forest, Naïve Bayes, Support Vector Machine and with three different variations of SVM which are SVM Linear Kernel, SVM RBF Kernel, SVM Sigmoid Kernel.

The evaluation section contains a critical role in providing performance indicators.

3.3 System Design

Use Case Diagram

A Unified Modelling Language (UML) diagram model is used to understand the structure and behaviour of the developing solution. This method aids in visualising and comprehending the architecture design of the implemented code and understand its functions. The user case diagram and activity diagram are the two diagrams that are used. Furthermore, the provided diagrams can be used for future work to implement the delivered solution as a software system with user interaction.

Figure 3.1 illustrates a use case diagram that represents the possible applied models that are used as a solution for this project. The five available options involve three different classification models types and three different variations of one of the models. All classifiers models are as follow(Random Forest, Naïve Bayes, SVM Linear Kernel, SVM RBF Kernel, and SVM Sigmoid Kernel).

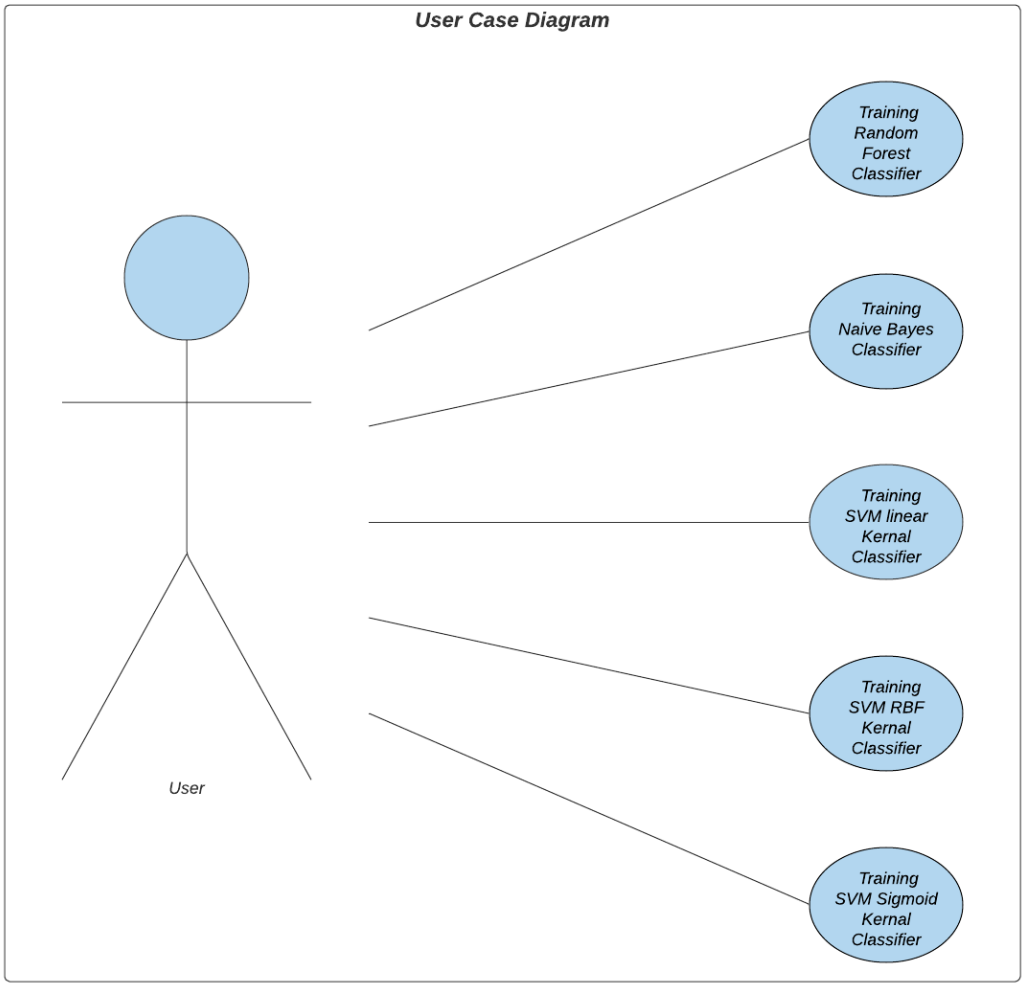


Figure 3.1 User Case Diagram

In addition, Figure 3.2 demonstrates the approach which contains the behaviour aspects of the solution implemented as an activity diagram. The following flowchart includes the steps taken after raw data is loaded to the script, which are reading data, data exploration (EDA) pre-process data involves the following steps which are; divides the data into a training and test set, feature selection for each model relies on the model type for each model and classifies the data as malicious or benign through use of the available classification models. Finally, after predicting the test data, the evaluation metrics process is conducted using different approaches of calculating the results and comparing them with the outcome of various classification models.

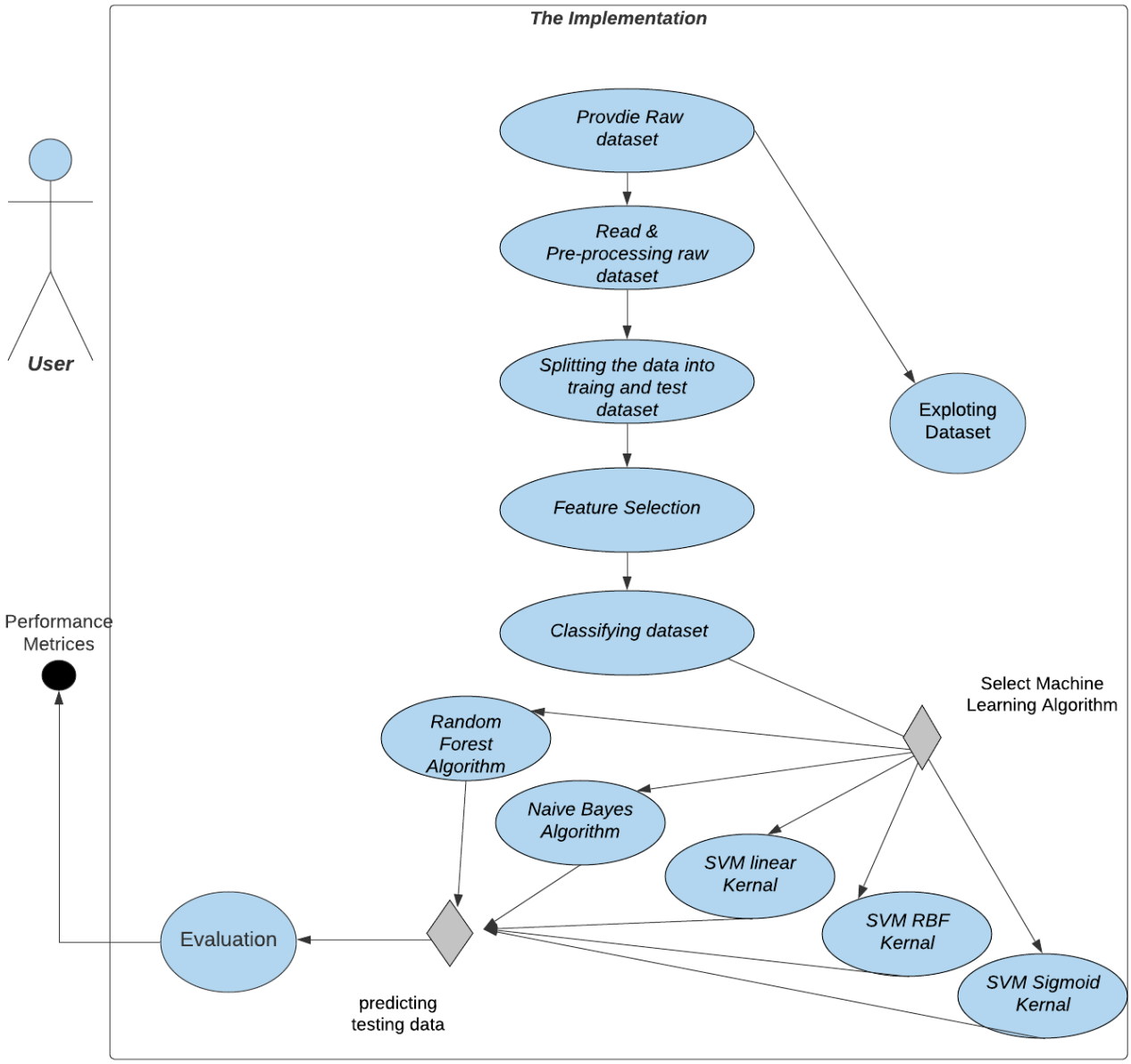


Figure 3.2 Use Case Diagram : Classification system

3.4 Development Methodology

Due to the limited time frame in which the solution framework had to be developed and tested, a development strategy also called methodology must be followed to manage that all deliveries were achieved by the project end, the development and testing of the system design and to ensure delivering an optimal project result. Therefore, In this project, the Agile software development methodology has been selected to be the appropriate method and was used during the implementation and testing phase. The agile development method splits the project into small incremental builds called

iterations. At the end of every iteration, a working product is presented to the client. Here are various reasons for applying the agile methodology, why Agile was chosen over other methods in this project. Firstly, the requirements for this project are at a moderate to high risk of changing. Agile is a suitable method to be applied to this project, since it In allows editing, modifying or even changing the requirements, which can be incorporated at any point of the process even in late development processes without the risk of losing the entire work. Secondly, a working software will be delivered considerably quicker and successive iterations can be delivered frequently and viewed by the client at a consistent pace. Thirdly, the client will be satisfied by the rapid and continuous delivery of the useful system, and also feedback will be obtained from the client after each iteration and can modify the project, which will provide the opportunity to improve the developed system in the next iterations. Though Agile is a more cooperative methodology with a team, planning it was easier for me with a single developer. I followed the approach by splitting down assigned work and the prioritisation process, considering feedback that received from my supervisor Amir after finishing each meeting, and finally by setting the planned objectives and adapt to the changing of the work to achieve the task within the specific time frame. Throughout the project, four iterations have been utilised, each with its own objectives and deliverables and are presented in the following list.

• Iteration One: The first iteration focused on implementing loading, parsing and exploring the data set provided. The deliverable from this iteration was pre-processed dataset that can be utilised in classification models.

• Iteration Two: The second iteration focused on implementing three classifiers and with three variations of one of these algorithms and allow them to be trained and tested using the provided dataset resulted from the first iteration. The deliverable from this iteration was the five classifiers being able to be trained and tested.

• Iteration Three: The third iteration focused on selecting features for model training. The deliverable from this iteration was choosing the features by using required feature selection methods; for example, feature importance and recursive feature removal functions.

• Iteration four: The last iteration focused on creating a set of different evaluation to measuring the result of each classifier and determine the best classification model. The deliverable from this iteration was comparing evaluation metrics for every machine learning classification model.

CHAPTER 4

IMPLEMENTATION

This section of the report will explain the implementation part.

4.1. Project Structure

In order, to produce a readable-friendly, reliable and maintainable malware detection system, python modular programming was implemented to structure the project by splitting the source code into separate multiple sections as mentioned previously. These sections are represented by utilising a python version 3.7, it is a powerful programming language and accessible most computer scientists use it as the primary language for presenting and developing data science project. Python is made up of classes, functions and variable definitions, that a complete system can be generated. Hence, the usage of modules facilitates the re-usability of the code and makes it easier to access the code for certain functionality. In addition, the code script is written in Jupyter notebook as it is an incredibly powerful tool for interactively developing, it is a flexible tool, one document where you can run and execute code. All the implementation section such as code execution, performance monitoring, and result visualisation in one single file. Jupyter network is technically easy to deal with since each part of the system works independently; therefore, this has the ability to test a specific block without executing the whole code from the beginning, as it will ease the access to specific functions.

The five parts demonstrated and discussed as follow:

1. Loading and Reading the CSV file.
2. Exploratory Data Analysis (EDA), which is an approach using descriptive statistics and graphical tools to better understand and analyse a given raw data sets and conclude their key attributes.
3. Pre-process data: Data preparation is a set of procedures that helps make the dataset more suitable for machine learning classifiers.
4. Machine Learning models (classifiers) contains three models; Random Forest, Naive Bayes and Support Vector Machine, which consists of three variations of kernel classification models. Also, it includes selecting the important features.
5. The Evaluation section contains an important role in providing performance indicators, as it is an important part of the project.

4.2. Data used

The dataset is a capture of the packets (**Pcap file**) converted into a Comma Separated Value (CSV) file of the dataset. The dataset was used in the analysis is a labelled malware dataset consisting of two types of network traffic, benign and malicious network traffic dataset. This dataset was captured by Adi, by running some malware and benign-ware on 15 computers, and then capturing all traffic from those computers using **Wireshark**. The dataset is a capture of the packets of network traffic in .pcap format which is converted into labelled a Comma Separated Value CSV file named "test_final labeled1", which contains all the important extracted features that would help in detecting the malicious network traffic. The **Pcap file** contains 11808 network traffic captured the first 30 minutes of the experiment and with a size of 808 MB.

4.3. Data Collection and Pre-processing

This part explains in depth the process of ingesting the dataset into a Python environment for processing and analysis.

4.3.1. Data ingestion

The first phase before starting the data processing step is fulfilled by importing the data collection into the Jupyter notebook. In a particular, the dataset utilized for analysis is in a specific format called "test_final labeled1.csv." The implementation snippet used to read the CSV file and to convert it to a data frame using a specific Python package named “Pandas” frame is illustrated in following Figure 4.1.

#Section 1: Loading a raw network traffic dataset

```
# Importing the dataset
import pandas as pd
dataa=pd.read_csv("test_final labeled1.csv")
```

Figure 4.1 Reading a csv file and converting it into Pandas Data frame

4.3.2. Data Description

For the purpose of this project, the Network traffic data set was labelled contains 20 columns (extracted features) of data with each column consisting of a header and 11808 rows. Each column of the CSV file contains valuable information about the captured network traffic, which are the features was extracted from the given **Pcap file**. Description of the columns of CSV file is summarised and explained in Table 4.1.

Field	Description
frame_number	The frame number
src_ip	Source IP: Source IP address
dst_ip	Destination IP : Destination IP address
Dur	Time of the last packet seen-Time of the first packet seen (Record total duration)
ip_proto	Assigned Internet Protocl Numbers 17 : UDP User Datagram Protocol 1 : ICMP Internet Control Message protocol 6 : TCP Transmission Control Protocol
ip_ttl	Time To Live (TTL) value.
ip_hdr_len	Header length IP : Indicates the **header length** in 32-bit words.
ip_pkt_len	Packet length IP: This is often just a small part of a file, webpage, or other transmission, since individual **packets** are relatively small.
ip_checksum_status	validate the checksums of IP protocol, it also known as redundancy checking. A **checksum** is a small-sized block of data derived from another block of digital data for the purpose of detecting errors that may have been introduced during its transmission or storage.
stream_index	Column displays a unique number for each stream, such as 1 for the first stream, 2 for the second stream, et cetera.
src_tcp	Source TCP : Source port number
dst_tcp	Destination TCP Port: Destination port number
tcp_seq	Sequence TCP Port : The **sequence number** is the byte **number** of the first byte of data in the **TCP** packet sent (also called a **TCP** segment).
timestamps	Timestamps
tcp_flags_ack	The acknowledgment flag is used to acknowledge the successful receipt of a packet.
tcp_flags_syn	The synchronisation flag is used as a first step in establishing a three way handshake between two hosts.
tcp_flags_fin	The finished flag means there is no more data from the sender. Therefore, it is used in the last packet sent from the sender.
tcp_flags_urg	The urgent flag is used to notify the receiver to process the urgent packets before processing all other packets. The receiver will be notified when all known urgent data has been received.
trafficCat	The name of each attack in string: Benign and malicious
Label-final	0 for normal and 1 for attack records

Table 4.1: Columns Description of all extracted features from pcap file

It is significant to take under consideration that not all the 20 columns are included in the analysis because there are some features that unrelated and not useful. Therefore, features that represented identify properties were removed (e.g. The source and destination IP addresses), Since they considered intrusion detectors and to ensure the model was not dependent on specific network configurations. Additionally, in this report, certain columns need to be eliminated before starting the implementation. The following deleted features are identified with the specifying the reason:

- Timestamps: the timestamp of each network traffic is not related, while duration is related. So, it is been deleted from the existing file.

There are in total fourteen features remaining for the analysis. Table 4.2 includes a list of all the remaining feature with more details about their functionalities in analysing. All remained features have been cleaned and pre- processed to be ready for the machine learning classifiers models; detailed steps will be illustrated and explained in the upcoming section (4.3.3 data cleaning and pre-processing).

Field	Description
Dur	Time of the last packet seen-Time of the first packet seen (Record total duration)
ip_proto	Assigned Internet Protocl Numbers 17 : UDP User Datagram Protocol 1 : ICMP Internet Control Message protocol 6 : TCP Transmission Control Protocol
ip_ttl	Time To Live (TTL) value.
ip_hdr_len	Header length IP : Indicates the **header length** in 32-bit words.
ip_pkt_len	Packet length IP: This is often just a small part of a file, webpage, or other transmission, since individual **packets** are relatively small.
ip_checksum_status	validate the checksums of IP protocol, it also known as redundancy checking. A **checksum** is a small-sized block of data derived from another block of digital data for the purpose of detecting errors that may have been introduced during its transmission or storage.
stream_index	Column displays a unique number for each stream, such as 1 for the first stream, 2 for the second stream, et cetera.
src_tcp	Source TCP : Source port number
dst_tcp	Destination TCP Port: Destination port number
tcp_seq	Sequence TCP Port : The **sequence number** is the byte **number** of the first byte of data in the **TCP** packet sent (also called a **TCP** segment).
tcp_flags_ack	The acknowledgment flag is used to acknowledge the successful receipt of a packet.
tcp_flags_syn	The synchronisation flag is used as a first step in establishing a three way handshake between two hosts.
tcp_flags_fin	The finished flag means there is no more data from the sender. Therefore, it is used in the last packet sent from the sender.
tcp_flags_urg	The urgent flag is used to notify the receiver to process the urgent packets before processing all other packets. The receiver will be notified when all known urgent data has been received.

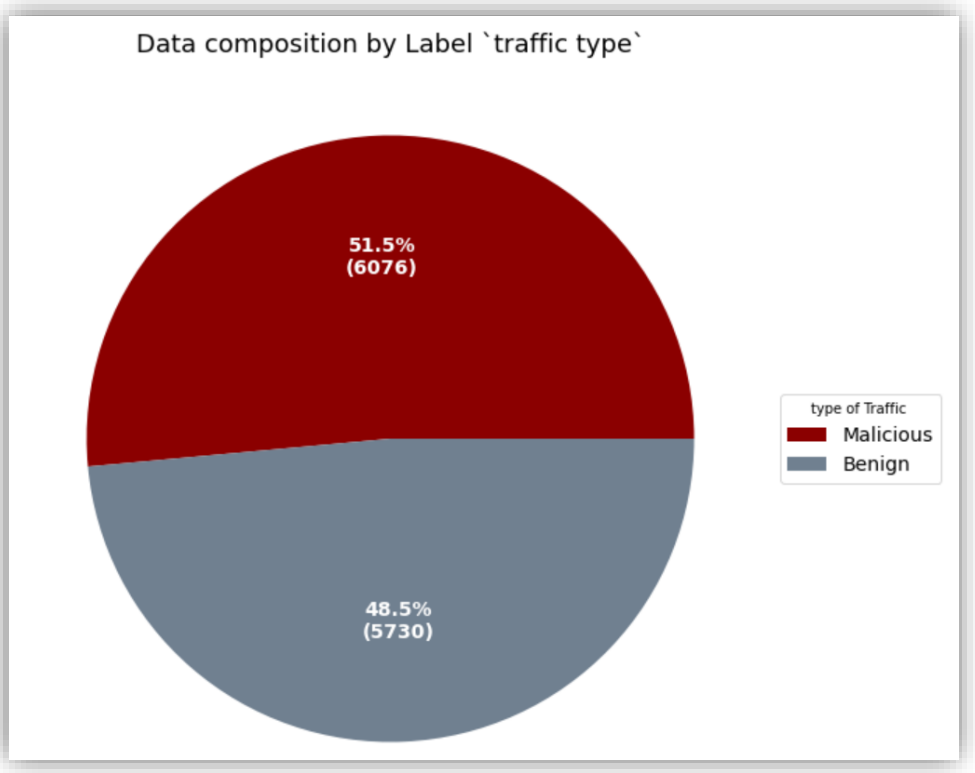
Table 4.2: Columns Description of features used

Several features have been used and prepared for the Machine learning models and were extracted. As it can be seen in the previous feature table 4.2 that most of the features have been utilized from IP and TCP packets since TCP and IP headers contains important features to detect network traffic. Based on a study that has been made that several experiments results illustrated that malicious traffic can be recognised from legitimate ones by observing only TTL values in the IP packet [46]. Another important feature is the IP packet length, where Packet length is also playing a significant role as an indicator of malicious behaviour, specifically when the packet is significantly larger or smaller than usual. The destination port of a packet is another useful feature for detecting activity such as port scanning which generally involves several probes to one or more ports. Moreover, using TCP flags are the most important features to be used in the Machine learning approach to detect malicious network traffic. According to research indicates that various combinations of TCP flags are considered as a crucial indicator of malicious activity [6]. Also, another essential feature in the TCP packet is the TCP Sequence. Network packets hold useful information about network activities since important data can be retrieved from them such as packet streams (e.g. TCP stream index) that can be used as a significant feature to detect malicious network traffic and behaviour [28].

4.3.3. Exploratory Data Analysis and Visualisation

Exploratory Data Analysis is an important part of any data analysis, it provides numerical detective work to gain more in-depth details of the data using histograms, charts drawn and tables from the data [15].In this context, the visualization technique of EDA was essential to be applied to the existing datasets for learning and prediction. In this report work, to achieve this goal various graphical techniques are used for a better visual understanding. Based on the results obtained it can be found that EDA plays a significant role in describing the data using statistical and visualization analysis without making any speculations about the content. With this dataset and EDA approach, we have achieved a clear understanding of the significant features needed to predict if the network traffic is benign or malicious. This section will explain and show several summary statistical and graphical representation of variables.

- Sample size: we have a labelled dataset consisting of network traffic and whether they are malicious or benign, thus Figure 4.2 illustrates the number of records in the dataset is 11808 with 2 distinct labels – Benign and Malicious comprising 48.5% and 51.5% of rows respectively. Figure 4.2 display the Composition of the data by label "Traffic Type".



-

Figure 4.2 Data structure by traffic type

- Displaying data types: An important step in getting to know your data is to discover the different data types it contains. The columns of a Data Frame contain values of a specific data type Variable type (categorical or numeric) identification: To display all columns and their data types sing Python's Pandas' .info() method. Figure 4.3 in the Data Description section summaries the data type of the raw file, where pandas use the NumPy library to work with these types. Figure 4.3 shows all CSV file columns.

```
In [7]: # View- of the columns
        dataa.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11808 entries, 0 to 11807
Data columns (total 19 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   frame_number        11808 non-null  int64
 1   src_ip              11808 non-null  object
 2   dst_ip              11808 non-null  object
 3   dur                 11808 non-null  float64
 4   ip_proto            11808 non-null  int64
 5   ip_ttl              11808 non-null  int64
 6   ip_hdr_len          11808 non-null  int64
 7   ip_pkt_len          11808 non-null  int64
 8   ip_checksum_status  11808 non-null  int64
 9   stream_index        11808 non-null  int64
 10  src_tcp             11808 non-null  int64
 11  dst_tcp             11808 non-null  int64
 12  tcp_seq             11808 non-null  int64
 13  tcp_flags_ack       9702 non-null   float64
 14  tcp_flags_syn       9702 non-null   float64
 15  tcp_flags_fin       9702 non-null   float64
 16  tcp_flags_urg       9702 non-null   float64
 17  TrafficCat          11808 non-null  object
 18  Label-final         11808 non-null  int64
dtypes: float64(5), int64(11), object(3)
memory usage: 1.7+ MB
```

Figure 4.3 Data Type of the CSV file columns

- For Validation purpose, As it can be seen that the TCP flags contains some null values and to prepare for the machine learnings it is important to fill all the null values with valid values. Figure 4.4 illustrates the columns that contain null values.

```
In [8]: ## checking for all the null values
        dataa.isnull().sum()

Out[8]: frame_number            0
        src_ip                  0
        dst_ip                  0
        dur                     0
        ip_proto                0
        ip_ttl                  0
        ip_hdr_len              0
        ip_pkt_len              0
        ip_checksum_status      0
        stream_index            0
        src_tcp                 0
        dst_tcp                 0
        tcp_seq                 0
        tcp_flags_ack        2106
        tcp_flags_syn        2106
        tcp_flags_fin        2106
        tcp_flags_urg        2106
        TrafficCat              0
        Label-final             0
        dtype: int64
```

Figure 4.4 Checking for al null values

4.3.4. Data cleaning and Data pre-processing

A fundamental step of data analysis is to perform data cleaning and transformation to ensure the data is prepared and valid to be used for further use in machine learning classification models. The main objective of this project is to utilize the dataset for Machine Learning algorithms to be able to detect the labels accurately. So, that it can be used malware detection in real-time. In that regard, it is important that the data accuracy should be validated, column de-duplication (if any), blank values managed to be treated and processed, reformat values to useful data format and check the threshold values of the numeric fields.

Data cleaning and transformation steps are discussed here.

Data transformation: there are only one data binning and transformation instances in this project.

- Taking care of the missing data : Pandas automatically marks missing values or data with NaN (missing values), It is important to prepare the data for the machine learning Models by replace these NaNs with intelligent guesses such as the mean for the column as it shown in the Figure 4.5.

Taking care of missing data

In [14]:
```
#using mean
dataa['tcp_flags_ack'].fillna(int(dataa['tcp_flags_ack'].mean()), inplace=True)
dataa['tcp_flags_syn'].fillna(int(dataa['tcp_flags_syn'].mean()), inplace=True)
dataa['tcp_flags_fin'].fillna(int(dataa['tcp_flags_fin'].mean()), inplace=True)
dataa['tcp_flags_urg'].fillna(int(dataa['tcp_flags_urg'].mean()), inplace=True)
```

Figure 4.5 Taking care of the missing values

- Converting Continuous variables: Binning, also known as quantization is used for converting continuous numeric features into group data (categories) using quantile-based partitioning (binning), it is a good strategy to improve the efficiency of the classification algorithms such as Random Forest, Naïve Bayes and Support Vector Machine [7]. Figure 4.6 shows the conversion of continuous numeric variables

```
In [16]:
#Convert variables - Converting continious variables to bin numerical data into groups (Normalizing)
dataa['dur_cat'] = pd.qcut(dataa['dur'].rank(method='first'), q=20, labels=False )
dataa['ip_proto_cat'] = pd.qcut(dataa['ip_proto'].rank(method='first'), q=20, labels=False )
dataa['ip_ttl_cat'] = pd.qcut(dataa['ip_ttl'].rank(method='first'), q=20, labels=False )
dataa['ip_hdr_len_cat'] = pd.qcut(dataa['ip_hdr_len'].rank(method='first'), q=10, labels=False )
dataa['ip_pkt_len_cat'] = pd.qcut(dataa['ip_pkt_len'].rank(method='first'), q=10, labels=False )
dataa['ip_checksum_status_cat'] = pd.qcut(dataa['ip_checksum_status'].rank(method='first'), q=10, labels=False )
dataa['stream_index_cat'] = pd.qcut(dataa['stream_index'].rank(method='first'), q=20, labels=False )
dataa['src_tcp_cat'] = pd.qcut(dataa['src_tcp'].rank(method='first'), q=10, labels=False )
dataa['dst_tcp_cat'] = pd.qcut(dataa['dst_tcp'].rank(method='first'), q=10, labels=False )
dataa['tcp_seq_cat'] = pd.qcut(dataa['tcp_seq'].rank(method='first'), q=10, labels=False )
dataa['tcp_flags_ack_cat'] = pd.qcut(dataa['tcp_flags_ack'].rank(method='first'), q=10, labels=False )
dataa['tcp_flags_syn_cat'] = pd.qcut(dataa['tcp_flags_syn'].rank(method='first'), q=10, labels=False )
dataa['tcp_flags_fin_cat'] = pd.qcut(dataa['tcp_flags_fin'].rank(method='first'), q=10, labels=False )
dataa['tcp_flags_urg_cat'] = pd.qcut(dataa['tcp_flags_urg'].rank(method='first'), q=10, labels=False )
```

Figure 4.6 Conversion of continues numeric variables.

4.3.5. Train and Test Split

The machine learning model must be fitted to the training and test dataset as one of the most important requirements' training dataset usually more extensive than the test dataset, training set and testing set with having 75 % for training and 25 % for testing. It could be a powerful idea to split the data in a way, thus that the model can be trained on a larger portion in order to learn or adapt to more possible data constellations. Therefore, that it satisfies the statistical test and prevents sampling variability. Similar statistical properties should apply to both the training and test data set. The whole point of the train-test split is done to ensure a machine learning system is to be able to work with considerable accuracy even on unseen data.

- Training data for the model fitting
- Testing data for estimating the model's accuracy

In this project, there are three different Machine Learning algorithms, a test dataset with a 25% size of the sample is chosen for all the models. Train-test split function has been implemented to handle splitting the data by importing Sci-Kit learn from model_selection package. The code snipped to obtain this task is presented in Figure 4.7.

```
In [18]:
# Import train_test_split function from scikit learn
from sklearn.model_selection import train_test_split
y=dataa['Label-final'] # Labels
X=dataa[['dur_cat','ip_proto_cat','ip_ttl_cat','ip_hdr_len_cat','ip_pkt_len_cat','ip_checksum_status_cat','stream_index_cat',
'src_tcp_cat','dst_tcp_cat','tcp_seq_cat','tcp_flags_ack_cat','tcp_flags_syn_cat','tcp_flags_fin_cat','tcp_flags_urg_cat']]#F

# Split dataset into training 75% set and test set 25%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25) # 75% training and 25% test
```

Figure 4.7 Split dataset into Train-Test and Test Set

4.3.6. Feature Importance and Feature selection

Feature selection is considered as one of the significant pre-processing steps in building machine learning models for the detection of the malicious network [33]. The process of scoring the features to indicates the importance of the choice of feature selection method and to understand its relative importance amongst a large number of features is known as feature importance. The selection of features is a pre-processing phase utilised to minimise the number of features in building machine learning classification models. This pre-processing step is utilised to minimise the number of dimensions (features) in building a machine learning classification model without losing any significant information from the data. The feature selection allows the classifier model to perform faster training, decreases complexity, facilitates interpretation, and improves the accuracy of the classifier.

This project involves one feature importance and selection algorithm, which is the Random Forest algorithm.

As it can be seen in figure 4.8, it has been used Random Forest algorithm for feature importance implemented in scikit-learn as the RandomForestClassifier classes. After being fit, the model provides a feature_importances_ property which can be accessed

to retrieve an n-dimensional array with a normalized importance score, the position of an item in the array follows the index of the feature list [4].

Random Forest-based feature selection approach can also be used with the bagging and extra trees algorithms. Also, in the upcoming chapter "result and discussion" there will be discussed in more details of the result of this process.

```
[39]: # Run the Random Forest Classification model to get variable importance first
      from sklearn.ensemble import RandomForestClassifier

      #Create a Gaussian Classifier
      clf=RandomForestClassifier(n_estimators=100)

      #Train the model using the training sets y_pred=clf.predict(X_test)
      clf.fit(X_train,y_train)
      y_pred=clf.predict(X_test)
```

```
In [21]: # Feature selection - Print the variable importance
         import pandas as pd
         feature_imp = pd.Series(clf.feature_importances_, index=
         ['dur_cat','ip_ttl_cat','ip_proto_cat','ip_hdr_len_cat','ip_pkt_len_cat',
          'ip_checksum_status_cat','stream_index_cat','src_tcp_cat','dst_tcp_cat',
          'tcp_seq_cat','tcp_flags_ack_cat','tcp_flags_syn_cat','tcp_flags_fin_cat','tcp_flags_urg_cat']).sort_values(ascending=False)
```

Figure 4.8 Random Forest Feature Importance

4.3.7. Training and Evaluating the Model

Many machine learning algorithms are tested in most data science projects to identify the one and the best fits the data provided and is reasonably remains stable over repeated algorithm implementations. The repetition method increases the generalisation of the model and decreases the bias.

Three different Machine Learning algorithms were implemented in this project, each trained on the clean dataset with the most significant feature set particularly generated for the algorithm. This project premises on classifying network traffic into two classes Benign and Malicious using supervised machine learning classifications models; therefore, evaluating the machine learning classification algorithm as discussed in the background chapter, which is an essential part of any project. The **Scikit-learn** metric library was utilized to import the evaluation metrics functions to evaluate the performance of the models by comparing performance metrics for each classifier. It

includes Accuracy, Precision, and Recall, F1-Score, Area Under the Curve (AUC), Confusion Matrix, classification metric functions. The list below illustrates the performance metrics that were used from importing the **Scikit-learn** package with base class "metrics" function, Which produces different functions performance metrics functions and Figures 4.8 shows how they called in the code.

All the different performance metrics functions named as shown below [Scikit]:

sklearn.metrics.accuracy_score( ): In classification the accuracy classification score function computes subset accuracy.

sklearn.metrics.precision_score ( ): Calculates the precision, is the ability of the classifier to not label a negative class instance as positive

sklearn.metrics.recall_score ( ): Calculates the recall, is the ability of the model to find all the positive instances.

sklearn.metrics.f1_score ( ): Calculates the F1 score. The F1 score is the weighted average of precision and recall. Calculates the F1 score. The F1 score is the weighted average of precision and recall. The relative participation of precision and recall to the F1 score, which weights precision and recall equally

metrics.roc_auc_score ( ): Area under the Curve (AUC), Receiver Operating Characteristics curve (ROC), which computed from prediction scores.

These model performance metrics are created for each machine learning models and stored in a dataset. Generating a blank pandas data frame to store the specific performance metric for each machine learning models as illustrated in table 4.3. As it can be seen, The data frame contains 11 columns. The first column is generated to save the algorithm's name and the remaining columns are generated to store the metric described above, one score for the testing dataset and one for the training dataset.

```
[47]: # Creating blank dataframe to store the performance metrics
perf_metric = pd.DataFrame(columns = ["Model name","Accuracy - Train", "Accuracy - Test", "Precision - Train",  "Precision - Test", "Recall
perf_metric.head()
```

[47]:

	Model name	Accuracy - Train	Accuracy - Test	Precision - Train	Precision - Test	Recall - Train	Recall - Test	F1-Score - Train	F1-Score - Test	AUC - Train	AUC - Test
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Table 4.3 Blank Pandas Data Frame to store Performance metrics

Metric functions: Implementing the **sklearn**.metrics module to generate performance metrics to measure classification performance of the classifiers models and storing it in the empty Pandas data frame is shown in Figure 4.9.

```
#important model performance metrics that can be used to assess the model performance of a classification model.
#Import scikit-learn metrics module to calculate model accuracy
from sklearn import metrics
# Generating the performance metrics and storung in a dataframe PerfMetric
PerfMetric.iloc[0,0]='Random_Forest_Classifier'
#Accuracy
PerfMetric.iloc[0,1]=metrics.accuracy_score(y_train,y_pred_train)
PerfMetric.iloc[0,2]=metrics.accuracy_score(y_test,y_pred)
#Precision
PerfMetric.iloc[0,3]=metrics.precision_score(y_train,y_pred_train,average='binary',pos_label=1)
PerfMetric.iloc[0,4]=metrics.precision_score(y_test,y_pred,average='binary',pos_label=1)
#Recall
PerfMetric.iloc[0,5]=metrics.recall_score(y_train,y_pred_train,average='binary',pos_label=1)
PerfMetric.iloc[0,6]=metrics.recall_score(y_test,y_pred,average='binary',pos_label=1)
#F1 Score
PerfMetric.iloc[0,7]=metrics.f1_score(y_train,y_pred_train,average='binary',pos_label=1)
PerfMetric.iloc[0,8]=metrics.f1_score(y_test,y_pred,average='binary',pos_label=1)
#ROC Curve- AUC Score
PerfMetric.iloc[0,9]=metrics.roc_auc_score(y_train,y_pred_train)
PerfMetric.iloc[0,10]=metrics.roc_auc_score(y_test,y_pred)
```

Figure 4.9 Creating Performance metrics and storing in Data Frame

4.4. Classifier

In this project, three different types of machine learning classifier models have been developed with three different variations of one of the classifiers using classification algorithms imported from **Sklearn** classifier library which includes Random Forest, Naïve Bayes, Support Vector Machines. These models have been constructed as classes that independent of the other. The main aim of these models to be developed is to evaluate the performance of the machine learning classification models against the dataset, which have been selected based on their popularity and their success used in malicious network traffic detection.

The packages used for the implementation of algorithms are detailed as follows:

1. RandomForestClassifier is imported from **sklearn.ensemble** module.
2. GaussianNB classifier imported from **sklearn.naive_bayes** module.
3. SVC is imported from **sklearn.svm** module (Linear kernel, RBF kernel, Sigmoid kernel).

4.4.1. Implementing Random Forest Classifier

In this experiment, we run Random Forest Classifier as it is considered as is the supervised learning algorithm that is used in classifying network traffic into Benign or Malicious. According to the official documentation: "A random forest is a meta estimator that fits several classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting." For classification, applying random forest classifier model, RandomForestClassifier() class of the **sklearn.ensemble** library is used, which takes n_estimators as a parameter.

n_estimators: number of trees in the forest

The Random Forest Classifier hyperparameters used in the experiment are n_estimators = 100, which is also the default to represent the tree numbers [32]. To test the performance across train and test datasets, the model is fitted on the training dataset, f a model fit to the training dataset also fits the test dataset well. Then the predictions will be generated for both the test and train datasets. Figure 4.10 illustrates the complete implementation of the steps.

```
In [28]:
# #Model 1: Running Random Forest Classification

# Running the final Random Forest Classification after removing unimportant variables
import time
start= time.process_time()

from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#StandardScaler "feature scaling"
#Train the model using the training sets y_pred=clf.predict(X_test)
# Training the Random_Forest_Classifier Classification model on the Training set
clf.fit(X_train,y_train)

# prediction on the training set
y_pred_train=clf.predict(X_train)

# prediction on test set
y_pred=clf.predict(X_test)
```

Figure 4.10 Implementation of Random Forest Model

4.4.2. Implementing Naïve Bayes Classifier

Naïve Bayes (NB) is the second type of supervised machine Learning model utilised in the project. The Naive Bayes statistical classification technique is based on the Bayes Theorem. It is one of the most straightforward supervised learning algorithms. The Naive Bayes classifier is considered as one of the faster classification models, accurate, and reliable algorithm. Naive Bayes classifier has high accuracy and speed on large datasets. To implement the Naive Bayes algorithm, Gaussian() is used for classification as shown in Figure 4.11[27].

```
In [30]:
#Training the Naive Bayes model on the Training set
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
#Create a Gaussian Classifier
clf = GaussianNB()

# Training the Naive_bayets Classification model on the Training set
#StandardScaler "feature scaling"
# Train the model using the training sets
clf.fit(X_train, y_train)

# prediction on the training set
y_pred_train=clf.predict(X_train)

# prediction on test set
y_pred=clf.predict(X_test)
```

Figure 4.11 Implementation of Naïve Bayes Model

4.4.3. Implementing Support Vector Machines Classifie

Support Vector Machines Classifier (SVC) is another type of supervised machine Learning model utilised in the project. The kernel selection, which defines the shape of the decision boundary, is the most significant hyper parameter for SVC [24]. Given the data structure, which is mostly categorical, it is important to experiment with non-linear kernels such as Sigmoid and Gaussian (RBF) kernels in addition to linear kernels. Therefore, the hyper parameter tuning that experimented in this project is entirely dependent on selecting the kernels. All other hyperparameters are left to their default values.

The implementation of the Support Vector Machines linear kernel classification is shown in Figure 4.12. Similar to the other classifiers, the model is fitted on the training dataset and

predictions are generated for the test as well as train dataset exclusively to test the performance across train and test dataset.

```
In [32]:
#Import SVM module from scikit learn
from sklearn import svm

#Fitting a linear kernel SVM Classifier

clf = svm.SVC(kernel='linear') # Linear Kernel

#StandardScaler "feature scaling"
# Training the SVM Classification model on the Training set
#Training the SVM model on the training data sets
clf.fit(X_train, y_train)

# prediction on the training set
y_pred_train=clf.predict(X_train)

#Predicting the response on test dataset
y_pred = clf.predict(X_test)
```

Figure 4.12 Implementation of Support Vector Machines classifier (Linear Kernel)

The implementation of the other two kernels (Gaussian (SVM RBF kernel) and Sigmoid kernel) are illustrated respectively in the following Figures 4.13 and 4.14.

In [28]:

```
# Model 3: Reunning Support Vector Machine Radial Basis Function (RBF) Kernel model
#
import time
start= time.process_time()



# RBF Kernel
clf = svm.SVC() # RBF Kernel (default option)

#Training the SVM model on the training data sets
clf.fit(X_train, y_train)

# prediction on the training set
y_pred_train=clf.predict(X_train)

#Predicting the response on test dataset
y_pred = clf.predict(X_test)
```

Figure 4.13 Implementation of Support Vector Machines classifier (RBF Kernel)

In [29]:

```
# Model 4: Reunning Support Vector Machine sigmoid Kernel model
import time
start= time.process_time()



# Sigmoid Kernel
clf = svm.SVC(kernel='sigmoid')

#Training the SVM model on the training data sets
clf.fit(X_train, y_train)

# prediction on the training set
y_pred_train=clf.predict(X_train)

#Predicting the response on test dataset
y_pred = clf.predict(X_test)
```

Figure 4.14 Implementation of Support Vector Machines classifier (Sigmoid Kernel)

CHAPTER 5

SYSTEM TESTING

To demonstrate the system that have been developed works as intended and satisfies the system requirements, test cases have been generated and carried out on Windows 10 operation system device in Anaconda3 navigator using Jupyter notebook.6.01 and Python version 3.9 to run the system.

5.1Test case

Test Case Id: 01	**Test Purpose:** Loading Raw Dataset, labels		
Preconditions: None			
Test Case Steps: 2			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	Input the name of the label file in the reading file cell ()	When file added. To check this run the view cell, all variables in the file are shown	Pass
Comments: None			
Related Tests: -			

Table 5.1: TC-1 Loading raw dataset, labels

Test Case Id: 02	**Test Purpose:** Exploring Data Analysing (EDA) procedure to explore and visualize the existed raw dataset		
Preconditions: A provided dataset that was previously loaded and read by running TC-1			
Test Case Steps: 3			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	Run the labelled dataset (Start of EDA) section to see the comparison of the data by labels of traffic type (malicious traffic and Benign Traffic) in the labelled data frame	A circle graph will appear to illustrates the number of benign and malicious traffic in the dataset.	Pass
Comments: The test case passed all steps			
Related Tests: TC-1			

Table 5.2: TC-2 Exploring Data Analysing (EDA)

Test Case Id: 03	**Test Purpose:** Pre-processing the data to prepare it for the machine learning classification models		
Preconditions: An existing dataset was provided previously by running TC-1			
Test Case Steps: 3			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	Run the drop the column cell by entering variable's names that you want to drop	All specified variables are removed from the data frame. To check, run the view cell, all specified variables are removed from the data frame	Pass
2	Run the taking care of the missing data (null values) cell, with entering variable's names that contain null values	All specified variables contain null values from the data frame. To check, run the view cell, all the columns of the specified variables are filled with values (Mean)	Pass

3	Run Convert variables cell to bin numerical data into groups (Normalizing), with entering variable's names that you want to convert	New columns added for each specified variable with normalized values. To check this run the view cell, all variables are shown	Pass
Comments: The test case passed all steps. All cells will be included in the third section of the Jupiter file. The section name is #Section 3: pre-processing step			
Related Tests: TC-1			

Table 5.3: TC-3 Pre-processing the dataset and prepare it for the classifiers

Test Case Id: 04	**Test Purpose:** Creating, Tartaning and testing Random Forest Classifier		
Preconditions: An existing dataset was provided previously by running TC-1, and pre-processing step TC-3			
Test Case Steps: 1			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	Running all cells under Model 1: Random Forest Classifier section	No error message appears	Pass
Comments: The test case passed all steps. All cells will be included in the 4th section of the Jupiter file. The section name is #Section 4: Machine Learning Models, under Model 1: Random Forest.			
Related Tests: TC-1 and TC3			

Table 5.4: TC-4 Creating and training Random Forest classifier

Test Case Id: 05	**Test Purpose:** Creating, training and testing Naïve Bayes
Preconditions: an existing data set that was previously saved to a data frame by running TC-1 and pre-processing step TC-3	

Test Case Steps: 1			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	Running all cells under Model 2: Naïve Bayes section	No error message appears	Pass
Comments: The test case passed all steps. All cells will be included in the 4th section of the Jupiter file. The section name is #Section 4: Machine Learning Models, under Model 2: Naïve Bayes			
Related Tests: TC-1 and TC-3			

Table 5.5: TC-5 Creating and training Naïve Bayes classifier

Test Case Id: 06	**Test Purpose:** Creating, training and testing Support Vector Machine (Linear kernel)
Preconditions: an existing data set that was previously saved to a data frame by running TC-1 and pre-processing step TC-3	
Test Case Steps: 1	

Step No	**Procedure**	**Response**	**Pass/Fail**
1	Running all cells under Model 2: Support Vector Machin – Linear kernel section	No error message appears	Pass
Comments: The test case passed all steps. All cells will be included in the 4th section of the Jupiter file. The section name is #Section 4: Machine Learning Models, under Model 3: Support Vector Machine – Linear kernel.			
Related Tests: TC-1 and TC-3			

Table 5.6: TC-6 Creating and training SVM Linear Kernel classifier

Test Case Id: 07	**Test Purpose:** Creating, training and testing Support Vector Machine (RBF kernel)

Preconditions: an existing data set that was previously saved to a data frame by running TC-1 and pre-processing step TC-3			
Test Case Steps: 1			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	Running all cells under Model 4: Support Vector Machin – RBF kernel	No error message appears	Pass
Comments: The test case passed all steps. All cells will be included in the 4th section of the Jupiter file. The section name is #Section 4: Machine Learning Models, under Model 4: Support Vector Machine – RBF kernel.			
Related Tests: TC-1 and TC-3			

Table 5.7: TC-7 Creating and training SVM RBF Kernel classifier

Test Case Id: 08	**Test Purpose:** Creating, training and testing Support Vector Machine (Sigmoid kernel)		
Preconditions: an existing data set that was previously saved to a data frame by running TC-1 and pre-processing step TC-3			
Test Case Steps: 1			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	Running all cells under Model 5: Support Vector Machin – Sigmoid kernel section	No error message appears	Pass
Comments: The test case passed all steps. All cells will be included in the 4th section of the Jupiter file. The section name is #Section 4: Machine Learning Models, under Model 5: Support Vector Machine – Sigmoid kernel.			
Related Tests: TC-1 and TC3			

Table 5.8: TC-8 Creating and training SVM Sigmoid Kernel classifier

Test Case Id: 09	**Test Purpose:** Feature Selection procedure for Random Forest

Preconditions: an existing data set that was previously saved to a data frame by running TC-1, pre-processing step TC-3, and Random Forest model TC-4			
Test Case Steps: 1			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	Identify the most important **features**.	No error message appears	Pass
2	Running feature selection cells within Model 1: Random Forest	A graph is shown to visualize variable importance	Pass
3	Create a new 'limited featured' dataset containing only those **features**.	No error message appears	Pass
4	Train a second classifier on this new dataset with the newly selected feature	No error message appears	Pass
Comments: The test case passed all steps. All cells will be included in the 4th section of the Jupiter file. The section name is #Section 4: Machine Learning Models, within Model 1: Random Forest.			
Related Tests: TC-1, TC-3, and TC-4			

Table 5.9: TC-9 Creating and Applying feature importance for Random Forest classifier

Test Case Id: 10	**Test Purpose:** Calculating the performance results of classifiers

Preconditions: an existing data set that was previously saved to a data frame by running TC-1 and pre-processing step TC-3, Finally, all ML models step TC-4, TC-5, TC_6, TC-7, and TC,8			
Test Case Steps: 7			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	Run performance metric table cell to see accuracy results for all classifiers	Table shows with accuracy scores for all classifiers	Pass
2	Run accuracy plot cell to see accuracy results for all classifiers in a graph	A bar graph shows with accuracy scores for all classifiers	Pass
3	Run Precision plot cell to see accuracy results for all classifiers in a graph	A bar graph shows with precision scores for all classifiers	Pass
4	Run Recall plot cell to see recall results for all classifiers in a graph	A bar graph shows with recall scores for all classifiers Pass	Pass
5	Run F1-score plot cell to see F1-score results for all classifiers in a graph	A bar graph shows with F1-score for all classifiers	Pass
6	Run AUC plot cell to see AUC score results for all classifiers in a graph	A bar graph shows with AUC scores for all classifiers	Pass
7	Run Confusion Metrics cell to see confusion metrics results for all classifiers	A confusion metrics plot shows	Pass
Comments: all these cells will be included in the last section of the Jupiter file. Section name is #Section 5: Evaluation.			
Related Tests: TC-1, TC-3, TC-4, TC-5, TC-6, TC-7, TC-8 and TC-9			

Table 5.10: TC-10 Measuring the performance results of all classifier

CHAPTER 6

RESULTS AND DISCUSSION

It is important to critically analyse the performance of the models after the successful execution of the three machine learning classifiers with three different variations of one of the classifiers. Therefore, this chapter of the document evaluates the different Machine Learning classification models to identify the best performing Machine learning model as a classifier, since the result of classification algorithms is to predict labels. Additionally, to evaluating the final result, feature importance is analysed and compared. These two aspects have been shown and discussed in this chapter.

6.1 Classification Models Performance

First, it is significant to the identity performances of each classifier model is crucial for ensuring that the selected model is working optimally and outputting effective insights and results as an intended classifier. In addition, this chapter display the importance attributed to each input variables as identified by the non-parametric Random Forest algorithm followed by evaluation of performance metrics were generated for each classifier. The models were trained using 75 % of the training dataset and were tested with 25% of the testing dataset and tested models were evaluated for their performances and effectiveness in detecting malicious network traffic. The Performance metrics were generated to compare and evaluate the performances of the different classifier models for the classification task, which includes accuracy, recall, precision, F1-Score and AUC. Classification accuracy and misclassification cases are combined by way of a confusion matrix.

6.1.1 Variable importance assessment

Variable importance refers to the extent to which a model relies upon a variable to make accurate predictions. Variables with high importance are drivers of the results and their values have a significant impact on the classifier outcome values. The more a model relies on a variable to make predictions, the more important it is for the model. Random

forest is the only algorithm in the experiment where variable importance calculated for it.

- In this paper a variable selection method, based on Random Forests and Support Vector Machines to determine the variable importance.

The random forest is a classification algorithm consisting of many decisions trees. In **scikit-learn**, feature importance is implemented, which measure the decrease in node impurity (weighted by the probability of reaching that node (which is approximated by the proportion of samples reaching that node)[12]. As mentioned in paragraph 2.2.1, the Gini impurity measure is one of the methods used in Random Forest algorithms to determine the optimal split from a root node, which is the probability of wrongly classifying a random item according to the distribution of the class labels of the dataset [9]. The decrease in Gini indicates higher variable importance and the higher the value, the more significant is the feature. The variable importance chart to the Random Forest Classifier has been shown in two following Figures 6.1 and 6.2.

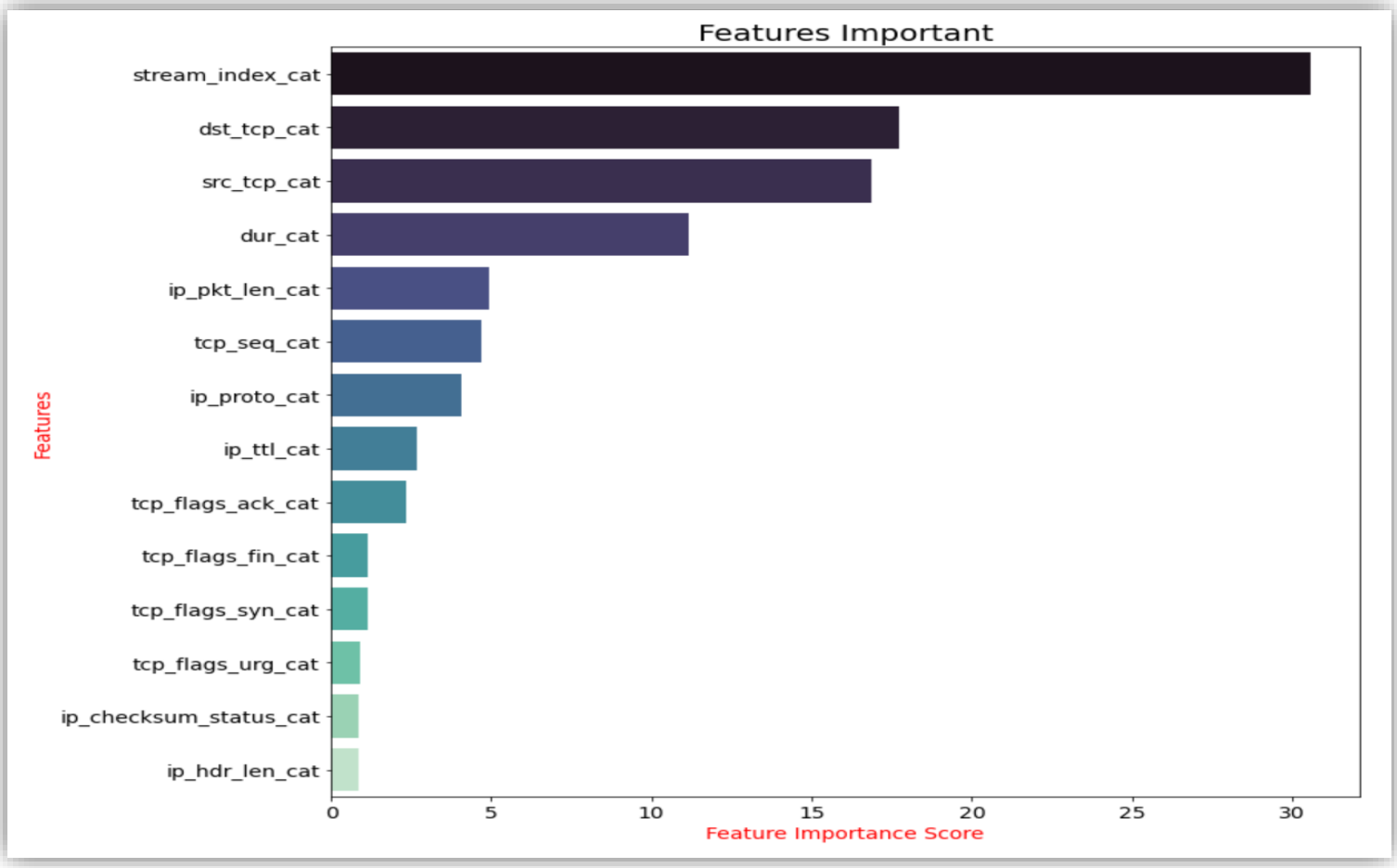


Figure 6.1 Variable importance From the Random Forest Classifier

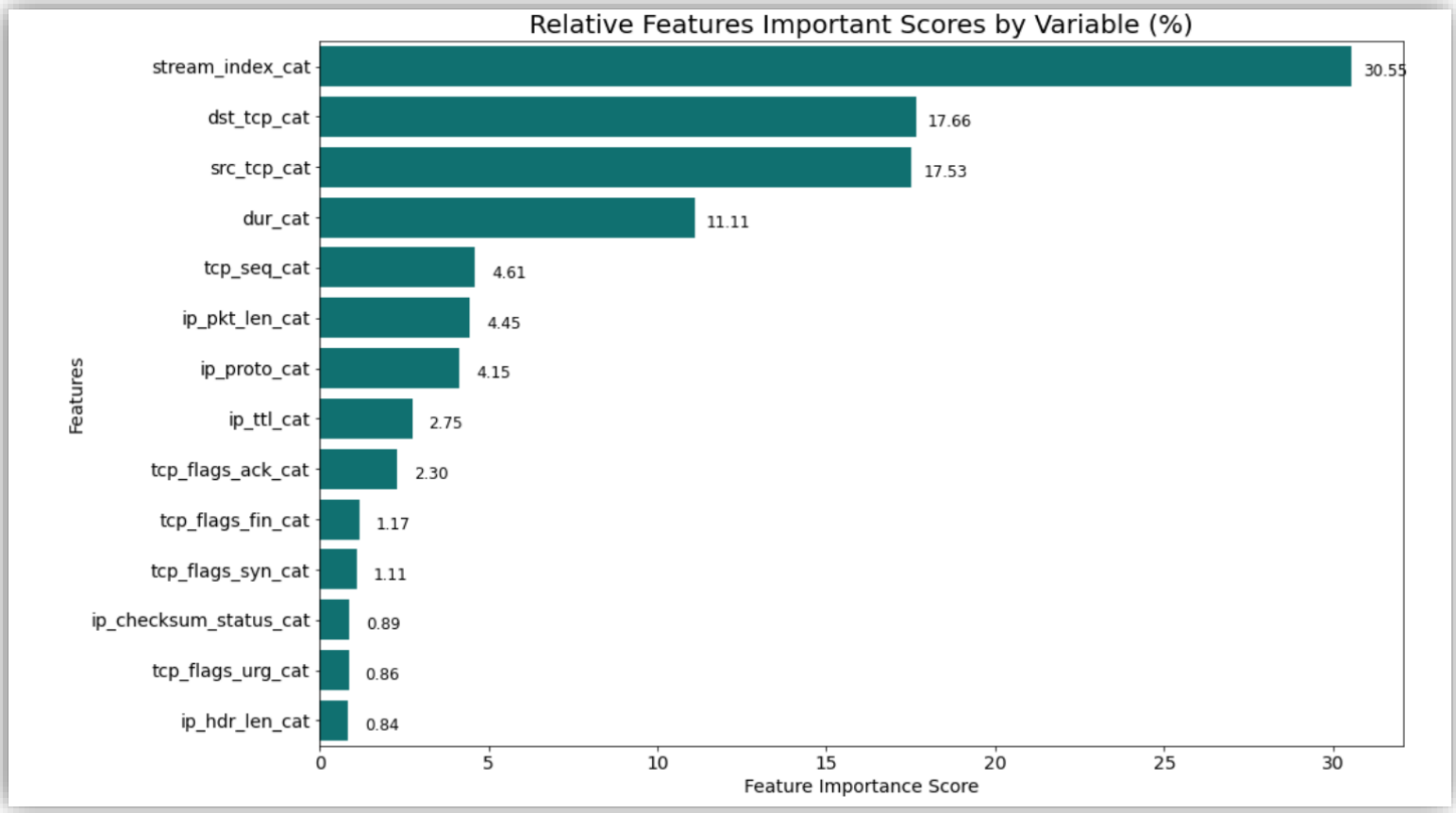


Figure 6.2 Variable importance scores by variable (%) From the Random Forest Classifier

Output format

Feature importance using Random Forest Classifier generates relative importance scores where all variables added to the new Random Forest classifier up to 1 or 100% score (refer to Figure 6.2). For example,

'stream_index_cat','dst_tcp_cat','src_tcp_cat','dur_cat','ip_pkt_len_cat','tcp_seq_cat','ip_proto_cat','ip_ttl_cat','tcp_flags_ack_cat','tcp_flags_fin_cat','tcp_flags_syn_cat', 'tcp_flags_urg_cat', 'ip_hdr_len_cat','ip_checksum_status_cat'

,which is scores 0.84 so it considered as an important feature to Random Forest Classifier.

6.1.2 Accuracy of the classifier

Figure 6.3 has been generated with the accuracy scores for the train and test data set that was obtained from testing the classifiers in several different tries. Accuracy is defined as the fraction of correct predictions amongst the total number of cases. It seems that Random Forest Classifier (RF) outperforms both Naïve Bayes (NB) and Support Vector Machine (SVM) and the other two different SVM models in regards to the accuracy scores. Random Forest scored an accuracy score is 99.53% for the train dataset and 91.6% for the test dataset, which is considered the highest among the classifiers. In addition, it can be noticed that all the classifiers perform well as classifiers on the selected dataset. The second highest accuracy scores recorded by SVM RBF kernel algorithms, which scored an accuracy score between 81.45% and 75.0%. While SVM Sigmoid Kernel performed abysmally and achieved an accuracy of around 50.45% in the train and 56.08% in the test dataset. Although the Sigmoid kernel is widely used, its properties are not fully studied [22].

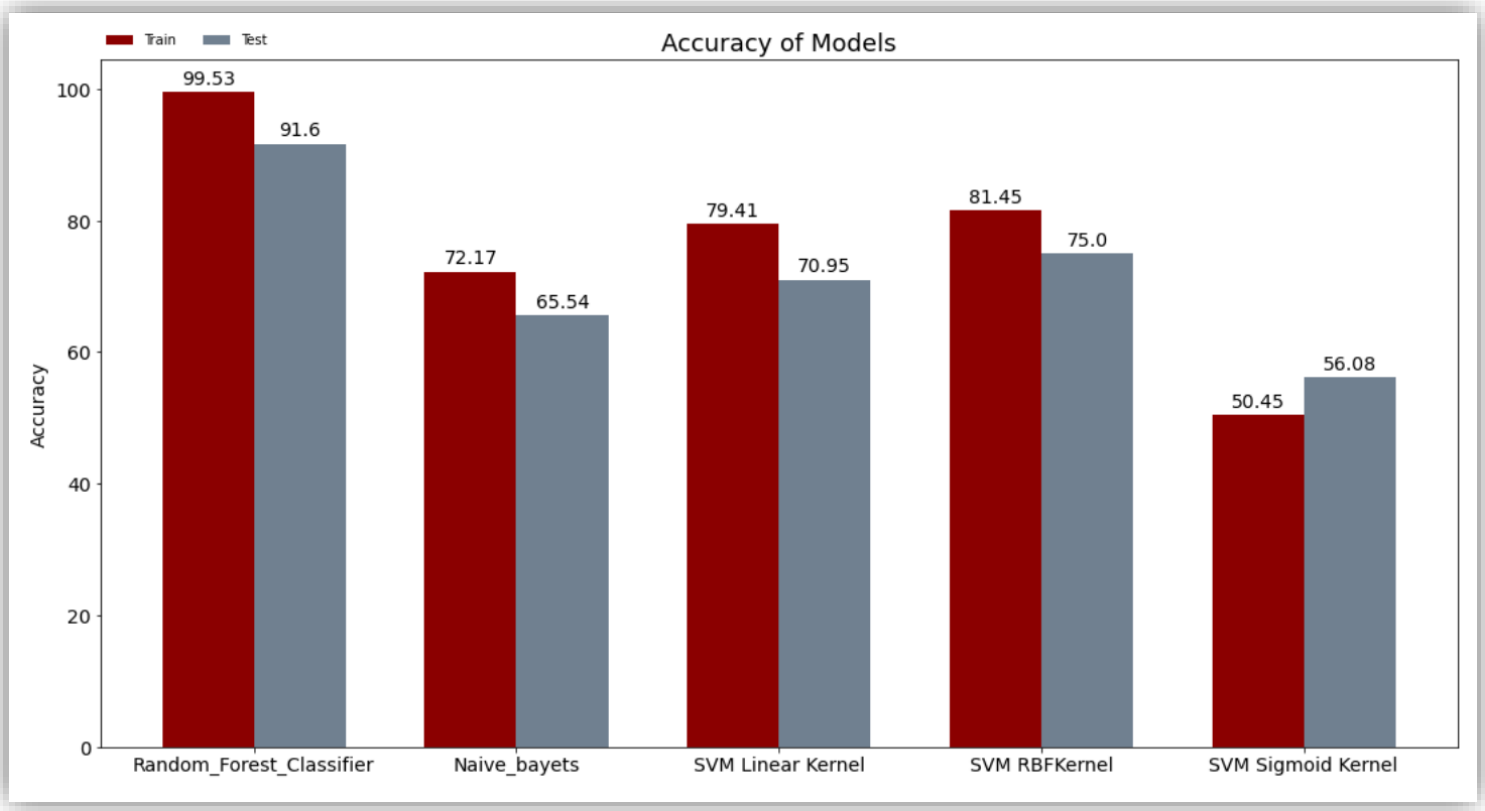


Figure 6.3 Comparison of accuracy scores by models

6.1.3 Precision of the classifier

Precision, as mentioned previously, is the ability of the classifier model to not labelling a negative class sample as correct positive and is computed by dividing the number of correct positive classifications by the total number of true positives and false positives. The outcome is 0 for no precision and 1 for is the best value, which is the highest value for full or perfect precision. [15].

Analysis of precision score suggests that although most models achieved from 76% to 50% precision scores for detecting malicious network traffic for the selected dataset, Random Forest (RF) outperformed other classifiers in the train set as well as in the test set. As it can be seen that Random Forest classifier scored between 99.47% and 90.02%. As a result, high precision here means a lower chance of labelling benign traffic as malicious. The precision scores of the classifier models are presented in Figure 6.4.

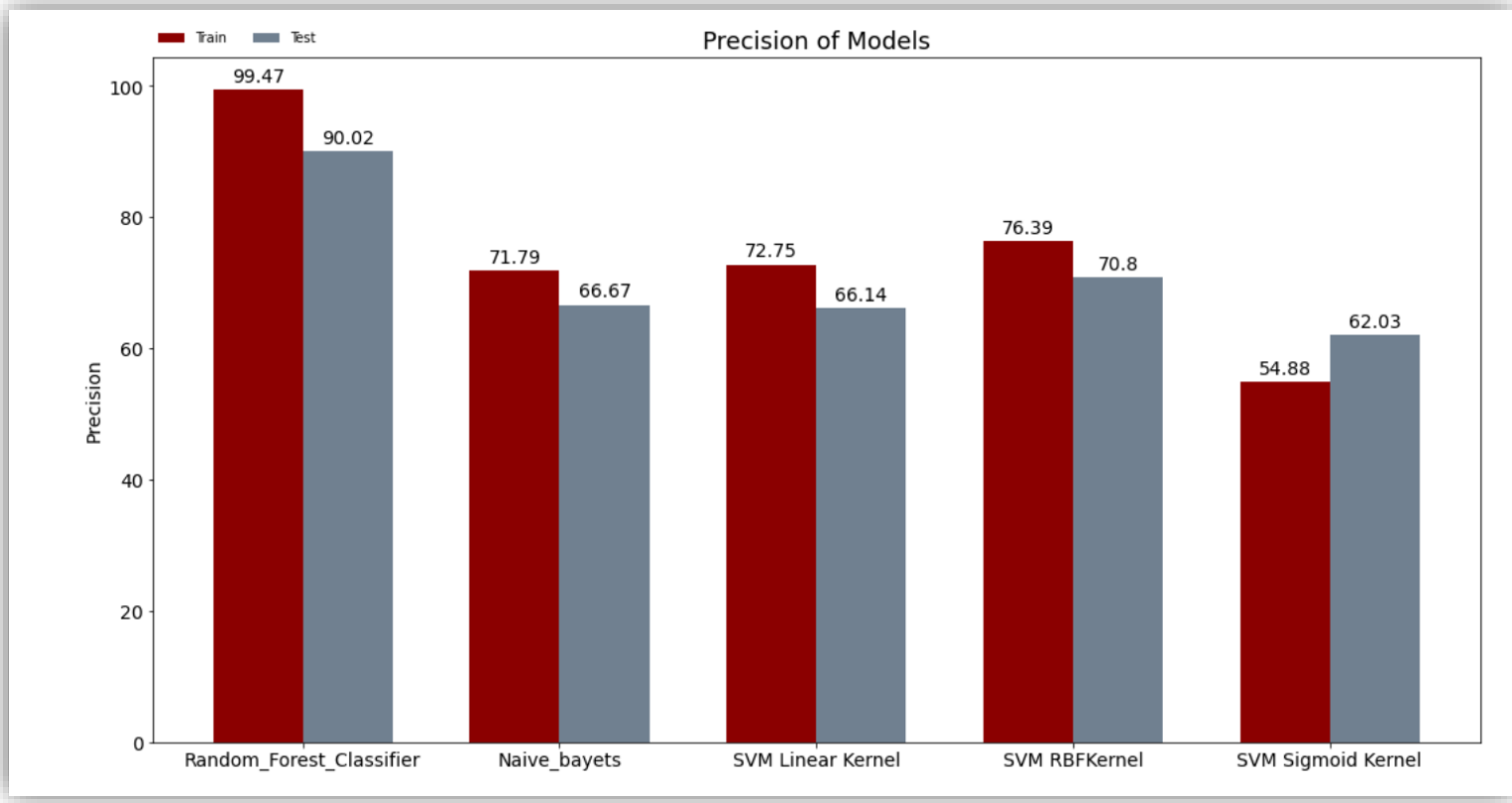


Figure 6.4 Comparison of Precision scores by models

6.1.4 Recall value of the classifier

The recall, as discussed previously, is the ability of the classifier to predict all the positive samples in the dataset. Therefore, Figure 6.5 illustrates the Recall scores of each classifier that were obtained from performing them, which it can be seen that almost all classifiers except SVM Sigmoid Kernel are doing well in classifying actual positive as positive in the chosen data set especially Random Forest and SVM Linear Kernel, where both classifiers outperform the other Classifiers even in predicting the positive instances correctly. While SVM sigmoid seems to have the lowest recall score among the classifiers. In the context of the project, as machine learning classifiers are generated to classify malicious network traffic, which can find positive labels are used to flag malicious traffic and hence high recall means that chances of not flagging actual malicious traffic as malicious is very low. Figure 5.5 compares recall score across the classification model.

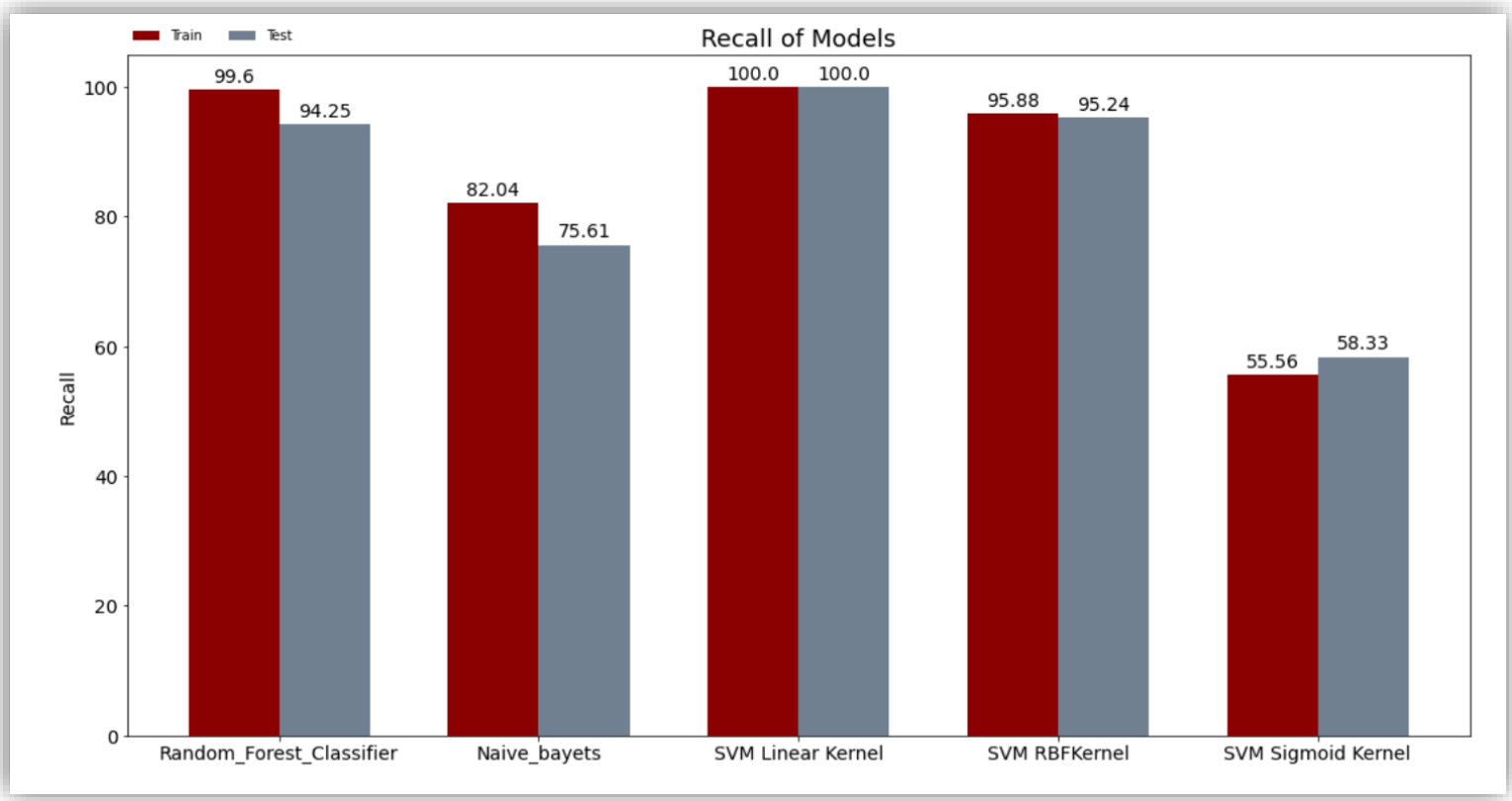


Figure 6.5 Comparison of Recall scores by models

6.1.5 F1-Score of the classifier

F-Score and also called F measure is a combined value between precision and recall, where the better value of F1 is 1, and the poor value is 0. It conveys the balance between the precision and the recall, where the contribution of precision and recall to the F1 score are equal. Higher F1-Score means a higher chance of predicting actual positives as positives and a low probability of predicting real negative as positive. According to this project experiment, it can be seen in the figure, that the Random Forest classifier models scored the highest F1-Score and other classification models achieved well enough F1 scores except SVM Sigmoid Kernel, which has the lowest F1-Score. Figure 6.6 compares F1-Score among various classifier models. For the purpose of this project, F1-Score considers as more important than accuracy because accuracy contributed by a large number of True Negatives and True Positive and both are not as important as F1-Score. False Negatives and False Positives are significant. An example of False Negative is when a file is malicious, and the classifier indicates it as not malicious, this could have a negative consequence [37].

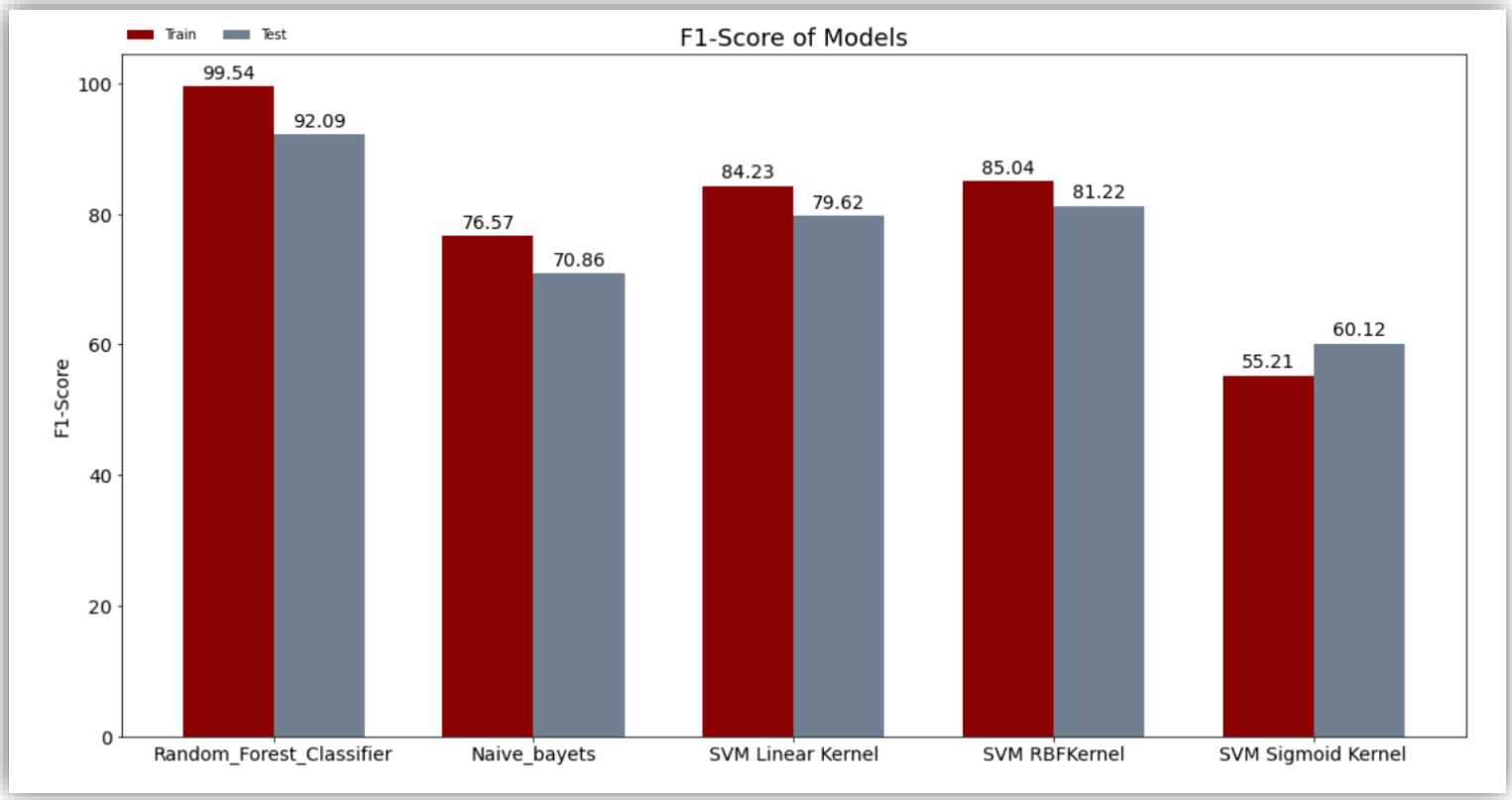


Figure 6.6 Comparison of F1- score by models

6.1.6 AUC of the classifier

As previously discussed, AUC is the area under the ROC curve. The True Positive Rate and False Positive Rate plots are the two parameters plotted on this curve [8]. As a result, ROC is the curve and AUC is the measure of separability, which indicates the capability of the classifier to distinguish between classes. The higher the AUC score is 1, which represent a better classifier. Although all of the models (except SVM Sigmoid Kernel) have an identical and high score of AUC score, Random Forest outperformed the other models in this regard. Figure 6.7 compares the AUC score across all the models.

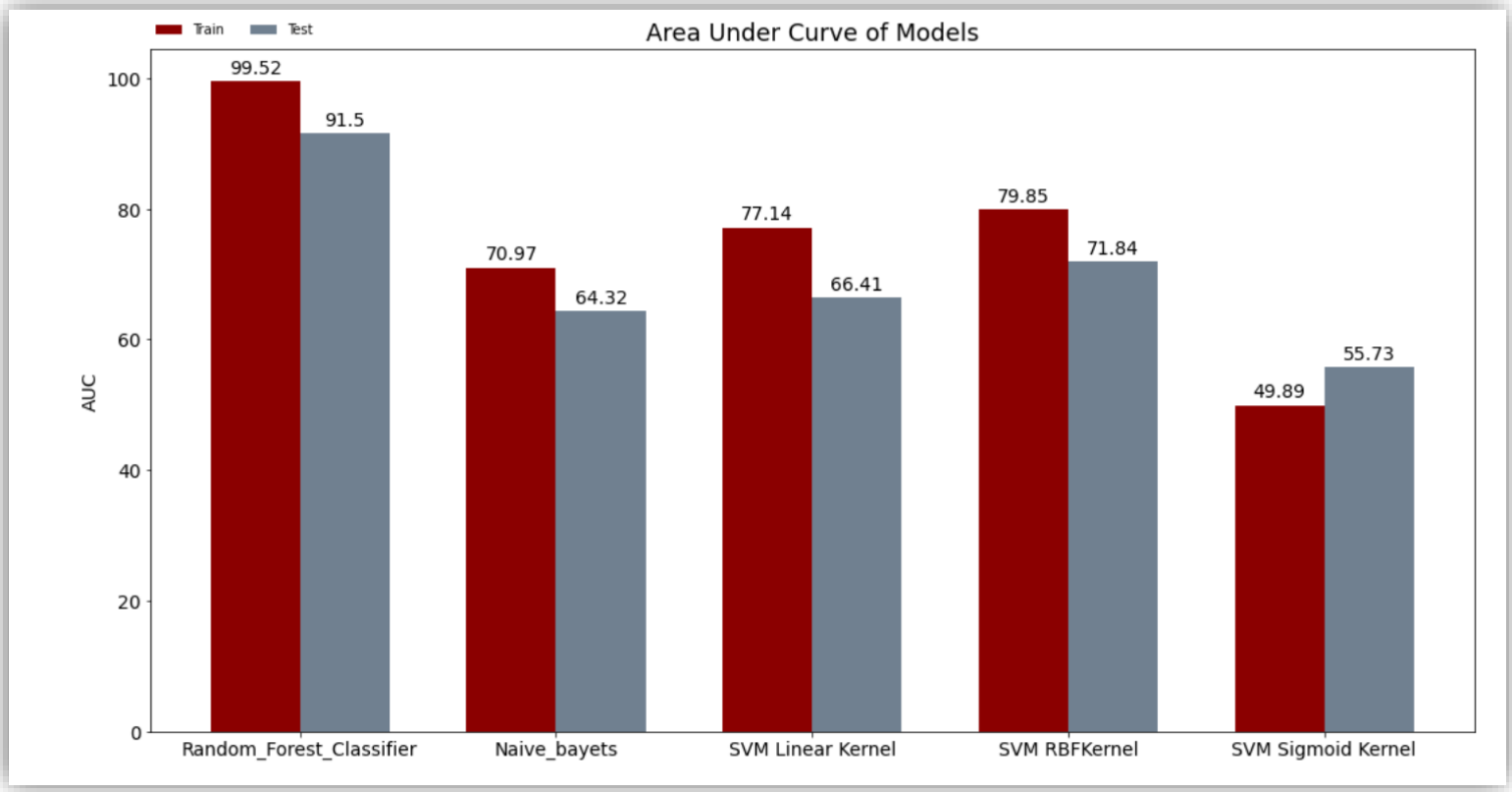


Figure 6.7 Comparison of Area Under Curve scores by models

6.1.7 Classifiers performance summary

A standalone comparison of performance metrics was generated to compare and evaluate the performances of the five different machine learning classification models that have been considered in the preceding section for the classification task. The models were trained with 75 % of the training dataset and were tested with 25% of the testing dataset and tested models were evaluated for their performances and effectiveness. In this section, all the performance metrics are summarised together in one place to identify the best performing model as shown in Table 6.1.

```
In [31]: # View the performance metric dataframe
         PerfMetric.head(6)
```

Out[31]:

	Model name	Accuracy - Train	Accuracy - Test	Precision - Train	Precision - Test	Recall - Train	Recall - Test	F1-Score - Train	F1-Score - Test	AUC - Train	AUC - Test
0	Random_Forest_Classifier	0.995257	0.915989	0.994728	0.900187	0.99604	0.942521	0.995384	0.920868	0.995236	0.914962
1	Naive_bayets	0.721719	0.655405	0.717857	0.666667	0.820408	0.756098	0.765714	0.708571	0.709696	0.6432
2	SVM Linear Kernel	0.794118	0.709459	0.727545	0.661417	1	1	0.842288	0.796209	0.771357	0.664062
3	SVM RBFKernel	0.81448	0.75	0.763934	0.707965	0.958848	0.952381	0.850365	0.812183	0.798519	0.718378
4	SVM Sigmoid Kernel	0.504525	0.560811	0.54878	0.620253	0.555556	0.583333	0.552147	0.601227	0.498883	0.557292

Table 6.1: Performance metrics data frame of classifiers

Analysis of Table 6.1 illustrates that the Random Forest model outperforms all others classification models in terms of Accuracy, Precision, Recall, F1 score and AUC. In addition, each of the five machine learning classification models was implemented. The models were trained with 75 % of the training dataset and were tested with 25% of the testing set and test set models were evaluated for their performances and effectiveness as a classification model.

The training dataset and test dataset performance metric of RF are also performed better than others in most cases. Visual inspection of three confusion matrix as represented in Table 6.2 for Random Forest classifier, best fit SVM model- SVM Linear Kernel and Naive Bayes, also illustrates that although the True Positive count is slightly higher than others, False-negative rate is very small. False Negative is costlier and more significant than false positive in Malware detection; hence, Random Forest and SVM performs satisfactorily.

Additionally, the Random Forest model can be considered as the only model that achieved and scored high for the majority of evaluation metrics and considerably represent the best performing classifier model; therefore, it considered as the most appropriate algorithm t solve malware detection problems.

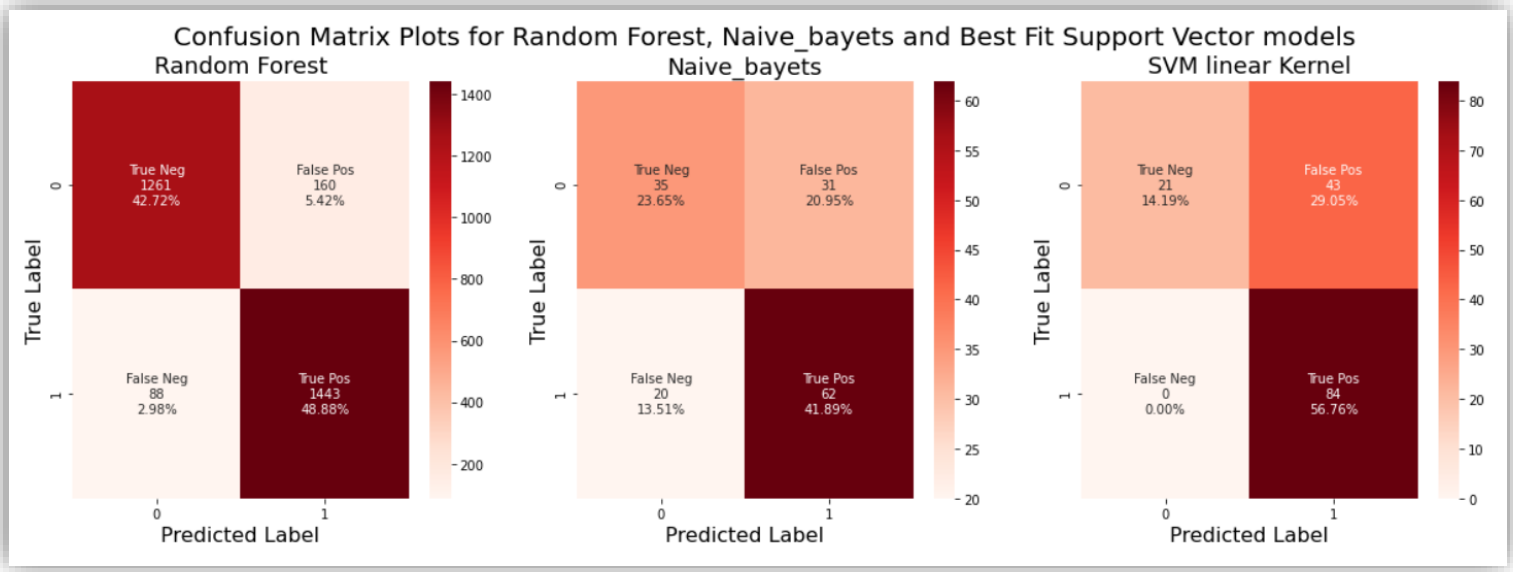


Table 6.2: Confusion matrix plots for the three best Models

In terms of the confusion matrix table for the Forest Random model which illustrates the classification prediction of the 25% testing data against the trained model with 75 % training data. Coherently, the accuracy of the Random Forest model as the classifier is satisfiable as to the prediction outcome results as indicating high for TP and TN and low for the FP and FN. Overall, it is clear from the results that the Random Forest classifier model has adapted the patterns of both malicious and benign network traffic and had proven to be the best performing classifier model.

6.1 Limitation

Due to the limited time allowed for implementing this project, there are several obstacles in achieving this project that will be discussed in this section. One of the main limiting factors that affected the quality, delivery, and overall project success is the coronavirus COVID-19 pandemic and its consequences of lockdown for months, the difficulty of travelling and backing home of international students. Another main challenge faced in this project that my laptop crashed during achieving the implementation which caused losing most of the project files that I worked on, where

affected my time, and effort toward achieving this project. Moreover, one of the challenges have been faced in doing this project of Machine learning to detection malicious traffic network field was the lack of resources on how the labelling process has been made. Also, since the dataset have been labelled manually, there will be a potential improper or erroneously labelling that could occur, which cause to incorrect processing of the classification output result and a significant threat to the entire scheme. Secondly, latest of the training dataset is also a strong consideration. The struggle with the complexity of malware that evolves and changes every day quickly is a never-ending battle; consequently practical utilise of the prototype can be assessed only on a real-time basis. Although challenges have occurred with malware detection projects, many solutions to the application of machine learning techniques for malware detection were applied and utilised due to its ability to keep pace with malware evolution as it is an interesting domain for researching.

CHAPTER 7

FUTURE WORK

Due to the limited time frame allowed to delivering this project, there are a variety of aspects of improving machine learning models and programming capabilities as well as the many opportunities to further the work that has been presented within this project that could potentially be improved in the future and to gain confidence. The first aspect would be to determine whether there are better fitting machine learning methods than those algorithms that have been utilized in this project, thus using more classifiers such as Neural Networks, Decision tree, Nearest Neighbour and Logistic Regression algorithms would be useful. The second aspect is enhancing the system that is to be built with a friendly user interface of the software system, which enables interaction with users. This could be done by utilizing web technologies such as, HTML, CSS combined with Java script to generate a graphical user interface that would make the system usable even by beginners. Another desirable improvement revolves around implementing the solution of the machine learning approach such as the use of a dataset that contains multi-class labels to make classification more specific, accurate and can be extended to identify not only malicious and benign network traffic, but also to categorise the type of attack occur. This would help to guarantee a more efficient classification to identify malicious traffic.

CHAPTER 8

CONCLUSION

8.1 Summary

To conclude, this project was aimed and set out at creating and utilizing a Machine Learning approach for classifying malicious and benign network traffic and each machine learning classifier will be examined to evaluate and identify the best model. In order, to achieve this, three several classifiers with three different variations of one of the implemented classifiers were performed. The solution that was applied to solve this certain problem is implementing Random Forest, SVM Linear Kernel, SVM RBF Kernel, SVM Sigmoid Kernel, and finally Naïve Bayes. Different performance metrics indicators were be utilized to test the effectiveness for each classifier, including accuracy, precision, recall, F1-score and AUC.

Based on the previous results were mentioned, all the Machine Learning models were achieved high values in different experiments except the SVM Sigmoid Kernel classification, which achieved the worst values, so it is considered a bad choice for solving a binary classification problem. In contrast, Random Forest achieved significantly the highest scores among all other relevant classifiers that were implemented. These are several key Success Factors for a Successful Implementation of Machine Learning algorithms such as; selecting the most appropriate algorithms for solving this specific type of problem in order to get the best result. Furthermore, multiple algorithms need to be evaluated and examined to ensure that the selected model fares the best among other relevant models. Also, a systemic approach for pre-processing data, selecting features and taking into consideration the dataset's quality was used in the experiment.

CHAPTER 9

REFLECTION

The final year project plays a very important role in the preparation for my professional life in the future. It was very significant to select a project in the final year according to my interest and passion and something that will help in my future life. I selected to work on Machine learning. I was very much interested in learning new things about the machine and especially machine learning related to the security field and solving security issues. This final project taught me how to understand the problem, the use cases and how to solve a certain problem and how to implement the best solution based on it. This experience was completely different from other university projects due to its value of benefits expected from the final project. There a large amount of effort, commitment and hardworking required to deliver to complete my graduation final project. One of the greatest lessons I learned by doing a final project, is the significance of writing the report alongside the time of implementing and writing the code, which required the perfect time management and how to manage my time between doing meetings with supervisor and completing the needed work. Besides these, I also gained a lot of new skills rather than just development. The machine learning domain expanded my knowledge in different aspects during achieving this final project, I learned about machine learning as a classifier. I learned how to write a clean and easy to understand code that will clarify itself in a very easy way. Moreover, I gained lots of useful information about the algorithms that have been implemented in this solution and their techniques and requirements such as; importing libraries, Load Dataset, pre-processing and Visualization, Building models. evaluating the training and testing data, Select Best Model, finally producing end-to-end a machine learning project and get a result, ending up with a complete piece of a report for the entire solution. Generally, I gained the opportunity to experience and got valuable knowledge from undertaking this project. Also, to enhance valuable skills; such as making-decision, analysing, problem-solving, research skills, management skills and overcome difficulties. All these skills that I have gained from working on this project will be utilised effectively in future job.

REFERENCES

1. Aldwairi, M., et al. 2021. (PDF) Detection of Drive-by Download Attacks Using Machine Learning Approach. [online] ResearchGate. Available at: https://www.researchgate.net/publication/317064301_Detection_of_Drive-by_Download_Attacks_Using_Machine_Learning_Approach[Accessed 20 February 2021].
2. Asiri, S., 2021. Machine Learning Classifiers. [online] Medium. Available at: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623 [Accessed 23 February 2021].
3. Awad, M. and Khanna, R.(2015). Efficient learning machines: theories, concepts, and applications for engineer s and system designer s. [Online]. Available at: https://link.springer.com/content/pdf/10.1007%2F978-1-4302-5990-9.pdf [Accessed: 18 March 2021].
4. Beaulieu, K. and Dalisay, D., 2020. Machine Learning Mastery. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com[Accessed 15 March 2021].
5. Bhatnagar, R., (2018). February. Machine Learning and Big Data processing: a technological perspective and review. In International Conference on Advanced Machine Learning Technologies and Applications (pp. 468-478). Springer, Cham. DOI: 10.1007/978-3-319-74690-6_46
6. Boukhtouta, A.et al. 2013. Towards Fingerprinting Malicious Traffic. Procedia Computer Science, [online] 19, pp.548-555. Available at: https://www.researchgate.net/publication/257719909_Towards_Fingerprinting_Malicious_Traffic[Accessed 20 February 2021].
7. Bowne-Anderson, H., 2018. Feature Engineering With Kaggle Tutorial. [online] DataCamp Community. Available at: https://www.datacamp.com/community/tutorials/feature-engineering-kaggle [Accessed 01 March 2021].
8. Brown G. 2009. Ensemble learning. C. Sammut, G. Webb (Eds.) Encyclopedia of Machine Learning, Springer [Online]. Available at: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30164-8_252 [Accessed: 20 January 2021]
9. Chio, C. and Freeman, D., 2018. Machine Learning And Security: Protecting Systems With Data And Algorithms. 1st ed. Beijing, China: O'Reilly.
10. C. Parker, "An Analysis of Performance Measures for Binary Classifiers," 2011 IEEE 11th International Conference on Data Mining, Vancouver, BC, 2011, pp. 517-526, doi: 10.1109/ICDM.2011.21.
11. ElBachirElMoussaid, N. and Toumanari, A., 2014. Web Application Attacks Detection: A Survey and Classification. International Journal of Computer Applications, [online] 103(12), pp.1-6. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.800.3515&rep=rep1&type=pdf [Accessed 13 April 2021].
12. Eryk Lewinson, E., 2019. Explaining Feature Importance By Example Of A Random Forest. [online] Towards Data Science. Available at: https://towardsdatascience.com/explaining-feature-importance-by-example-of-a-random forest-d9166011959e [Accessed 12 March 2020].
13. Gibert, D., Mateu, C., Planes, J., The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, Journal of

Network and Computer Applications (2020), doi: https://doi.org/10.1016/j.jnca.2019.102526.
14. Harley D., & P. M. Bureau. (2008). Drive-by downloads from the trenches. In Proceedings of the 3rd International Conference on Malicious and Unwanted Software, MALWARE 2008 (pp.98-103). Fairfax, VI, 2008.
15. Ho Yu, C., 2010. Exploratory data analysis in the context of data mining and resampling. International Journal of Psychological Research, [online] 3(1), pp.9-22. Available at: https://www.researchgate.net/publication/50946368_Exploratory_data_analysis_in_the_conte xt_of_data_mining_and_resampling [Accessed 15 April 2021].
16. Hyo-Sik Ham and Mi-Jung Choi, "Analysis of Android malware detection performance using machine learning classifiers," 2013 International Conference on ICT Convergence (ICTC), Jeju, 2013, pp. 490-495, doi: 10.1109/ICTC.2013.6675404.
17. Kantardzic, M. (2011). DATA MINING Concepts, Models, Methods, and Algorithms. [Online]. Available at: https://doc.lagout.org/Others/Data%20Mining/Data% 20Mining_%20Concepts%2C%20Models%2C%20Methods%2C%20and%20Algorithms %20% 282nd%20ed.%29%20%5BKantardzic%202011-08-16%5D.pdf [Accessed: 03 March 2021].
18. Kantardzic, M. (2011). DATA MINING Concepts, Models, Methods, and Algorithms. [Online]. Available at: https://doc.lagout.org/Others/Data%20Mining/Data%20Mining_%20Concepts%2C%20Models%2C%20Methods%2C%20and%20Algorithms%20%282nd%20ed.%29%20%5BKantardzic%202011-08-16%5D.pdf[Accessed: 12 April 2021].
19. Kishore, K. R., Mallesh, M., Jyostna, G., Eswari, P. R. L., & Sarma, S. S. (2014). Browser JS guard: Detects and defends against malicious JavaScript injection-based drive-by download attacks. In Proceedings of the 2014 Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT) (pp. 92-100).
20. Kirasich, Kaitlin; Smith, Trace; and Sadler, Bivin (2018) "Random Forest vs Logistic Regression: Binary Classi cation for Heterogeneous Datasets," SMU Data Science Review: Vol. 1: No. 3, Article 9. Available at: https://scholar.smu.edu/cgi/viewcontent.cgi?article=1041&context=datasciencereview [Accessed: 16 April 2021].
21. Leita, C., & Cova, M. (2011). HARMUR: Storing and analyzing historic data on malicious domains. In Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (pp. 46-53). ACM, New York, NY, USA.
22. Lin, H. and Lin, C., n.d. A Study On Sigmoid Kernels For SVM And The Training Of Non-PSD Kernels By SMO-Type Methods. Taipei 106, Taiwan: National Taiwan University.
23. Liu, C. 2020. A Top Machine Learning Algorithm Explained: Support Vector Machines (SVM). . Available at: A Top Machine Learning Algorithm Explained: Support Vector Machines (SVM) - KDnuggets [Accessed 16 March 2021].
24. Liu, Z. and Xu, H., 2014. Kernel Parameter Selection for Support Vector Machine Classification. Journal of Algorithms & Computational Technology, [online] 8(2), pp.163- 177. Available at:

https://www.researchgate.net/publication/273366480_Kernel_Parameter_Selection_for_Supp ort_Vector_Machine_Classification [Accessed 21 March 2021].
25. Miller, S. (2019). Traffic Classification for the Detection of Anonymous Web Proxy Routing Available at: 2019MillerSAPhD.pdf (ulster.ac.uk) [Accessed 20 Jan 2021].
26. Najafabadi, M. (2017). Machine Learning Algorithms For The Analysis And Detection Of Network Attack. Available at: Machine learning algorithms for the analysis and detection of network attacks | fau.digital.flvc.org [Accessed: 13 March 2021].
27. Naive Bayes. Available at: https://scikit-learn.org/stable/modules/naive_bayes.html[Accessed: 14 Fabruary2021]
28. O.MK. Alhawi, J. Baldwin, A. Dehghantanha Leveraging Machine Learning Techniques for Windows Ransomware Network Traffic Detection. A. Dehghantanha, M. Conti, T. Dargahi (EDS.), Cyber ThreatIntelligence, Springer, Chan(2018), pp. 93-106, 10.1007/978-3-319-73951-9_5
29. Priya, M., Sandhya, L., & Thomas, C. (2013). A static approach to detect drive-by-download attacks on webpages. In Proceedings of the 2013 International Conference on Control Communication and Computing (ICCC), (pp. 298-303).
30. Ray, S. 2017. 6 easy steps to learn Naive Bayes algorithm with codes in python and R Available at: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/ [Accessed: 13 March 2021].
31. Rieck K., Holz T., Willems C., Düssel P., Laskov P. (2008) Learning and Classification of Malware Behavior. In: Zamboni D. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2008. Lecture Notes in Computer Science, vol 5137. Springer, Berlin, Heidelberg
32. Rogel-Salazar, J., 2017. Data Science And Analytics With Python. Boca Raton, FL: Taylor & Francis Group.
33. R. Wald, F. Villanustre, T. M. Khoshgoftaar, R. Zuech, J. Robinson, and E. Muharemagic. Using feature selection and classification to build effective 127and efficient firewalls. In Proceedings of the 15th IEEE International Conference on Information Reuse and Integration, IRI 2014, Redwood City, CA, USA, August 13-15, 2014, pages 850–854, 2014.
34. SALIAN, I., 2018. NVIDIA Blog: Supervised Vs. Unsupervised Learning. [online] The Official NVIDIA Blog. Available at: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/[Accessed 11 Jan 2021].
35. Saravanan, R. and Sujatha, P.(2018). A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification. 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 945-949. IEEE.
36. [36]Scikit-learn (NO DATE). [Online]. Available at: https://scikit-learn.org/stable/index.html [Accessed: 15 February 2021].
37. Shung, K., 2018. Accuracy, Precision, Recall Or F1?. [online] Medium. Available at: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 [Accessed 21 Jan 2021].
38. Shwartz, S. et al. (2014). Understanding Machine Learning: From Theory to Algorithms. [Online]. Available at: https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf [Accessed: 02 April 2021].
39. Takata, Y., Akiyama, M., Yagi, T. Hariu, T. & Goto, S. (2015). MineSpider: Extracting URLs from environment-dependent drive-by download attacks. In Proceedings of the

2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSAC) (pp. 444-449). Taichung, Taiwan.
40. [40]TURING, I.B.A., 1950. Computing machinery and intelligence-AM Turing. Mind, 59(236), p.433.
41. T. O. Ayodele, ‘‘Types of machine learning algorithms,’’ New Adv. Mach. Learn., vol. 3, pp. 19–48, Feb. 2010.
42. V. L. Le, I. Welch, X. Gao, & P. Komisarczuk. (2013). Detecting heap-spray attacks in drive-by downloads: Giving attackers a hand. In Proceedings of the 2013 IEEE 38th Conference on Local Computer Networks (LCN) (pp. 300-303).
43. W. Anna, Z. Yue, H. Yun-tao and L. I. Yun-lu, "A novel construction of SVM compound kernel function," 2010 International Conference on Logistics Systems and Intelligent Management (ICLSIM), Harbin, 2010, pp. 1462-1465, doi: 10.1109/ICLSIM.2010.5461210.
44. Waskom, M., 2020. Seaborn. [online] PyPI. Available at: https://pypi.org/project/seaborn/ [Accessed 1 April 2021].
45. Witten, I. et al. (2016). Data Mining: Practical Machine Learning Tools and Techniques. Elsevier.
46. Yamada, R. and Goto, S., 2013. Using abnormal TTL values to detect malicious IP packets. Proceedings of the Asia-Pacific Advanced Network, [online] 34(0), p.27. Available at:https://www.researchgate.net/publication/266884443_Using_abnormal_TTL_values_to_detect_malicious_IP_packets [Accessed 17 February 2021].
47. ZAHARIA, A., 2021. 300+ Terrifying Cybercrime & Cybersecurity Statistics [2021 EDITION]. [online] Comparitech. Available at: https://www.comparitech.com/vpn/cybersecurity-cyber-crime-statistics-facts-trends/ [Accessed 04 May 2021].