# MALWARE DETECTION BY USING TWO-LAYER STACKING SVM CLASSIFIER

RAJA NURUL AINI BINTI RAJA MOHD ANUAR

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

# MOBILE MALWARE DETECTION USING TWO-LAYER STACKING SVM CLASSIFIER

RAJA NURUL AINI BINTI RAJA MOHD ANUAR

This report is submitted in partial fulfillment of the requirements for the Bachelor of [Computer Science (Computer Network)] with Honours.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2021

**DECLARATION**

I hereby declare that this project report entitled

**MOBILE MALWARE DETECTION BY USING TWO-LAYER STACKING SVM CLASSIFIER**

is written by me and is my own effort and that no part has been plagiarized without citations.

STUDENT: [illegible] Date : 12 SEPTEMBER 2021

RAJA NURUL AINI BINTI RAJA MOHD ANUAR

I hereby declare that I have read this project report and found this project report is sufficient in term of the scope and quality for the award of Bachelor of [Computer Science (Software Development)] with Honours.

SUPERVISOR : [illegible] Date: 12 SEPTEMBER 2021

TS. NOR AZMAN MAT ARIFF

## DEDICATION

Special dedicated to my beloved parents and fried who have encouraged, guided and inspired me throughout this journey of education

To my helpful lecturer, thank you for the guidance from the beginning until the end of this final year project

# ACKNOWLEDGEMENTS

All praise to Allah SWT for giving me the strength to finish my PSM, I manage to complete this Projek Sarjana Muda 1 (PSM1) and Projek Sarjana Muda 2 (PSM2). I would like to thank Ts. Nor Azman Bin Mat Ariff for giving assistant to complete this project successfully. Without his guide, this report and project cannot be completed.

Special thanks to my family for the continuous support throughout this journey from day to day. I would like to thanks to all my friend for their time, concern efforts, and always encourage me when preparing this report. With all the help from involves parties, I manage to complete my report and project which is Mobile Malware Detection Using Machine Learning.

Last but not least, thanks to Universiti Teknikal Malaysia Melaka (UTeM) for the opportunity given.

## ABSTRACT

Mobile device usage increased with new high technology that attracted the attacker to launch the attack, such as mobile malware. Mobile malware is malicious that is loaded into the device system and causes damage. Malware has become more prevalent in recent years. Nowadays, there are many techniques available to detect this attack. In this research, N-gram with opcode sequence dataset was used, focusing on mobile malware detection model. The dataset undergoes feature selection which is Information Gain and Chi-square, to reduce irrelevant data and redundant data. Then, the classifier used in this project is Support Vector Machine (SVM) to develop a model. The developed model will test and verify the accuracy with the test set. The project is giving hope to produce a system that can detect mobile malware attacks.

# ABSTRAK

Penggunaan peranti mudah alih meningkat dengan teknologi tinggi baru yang menarik penyerang untuk melancarkan serangan, seperti perisian hasad mudah alih. Perisian hasad mudah alih berniat jahat yang dimuat ke dalam sistem peranti dan menyebabkan kerosakan. Perisian hasad semakin berleluasa dalam beberapa tahun kebelakangan ini. Pada masa kini, terdapat banyak teknik yang ada untuk mengesan serangan ini. Dalam penyelidikan ini, N-gram dengan dataset urutan opcode digunakan, dengan fokus pada model pengesanan malware mudah alih. Set data menjalani pemilihan ciri iaitu *Information Gain* dan *Chi-Square*, untuk mengurangkan data yang tidak relevan dan data yang berlebihan. Kemudian, pengklasifikasi yang digunakan dalam projek ini adalah Mesin Sokongan Vektor (SVM) untuk membangunkan model. Model yang dibangunkan akan diuji dan mengesahkan ketepatannya dengan set ujian. Projek ini memberi harapan untuk menghasilkan sistem yang dapat mengesan serangan perisian hasad mudah alih.

# TABLE OF CONTENTS

# LIST OF TABLES

**PAGE**

# LIST OF FIGURES

## LIST OF ABBREVIATIONS

| | | |
|---|---|---|
| **FYP** | **-** | **Final Year Project** |
| **IT** | **-** | **Information Technology** |
| **OS** | **-** | **Operating System** |
| **ML** | **-** | **Machine Learning** |
| **AI** | **-** | **Artificial Intelligence** |
| **IG** | **-** | **Information Gain** |
| **CS** | **-** | **Chi-Square** |
| **NLP** | **-** | **Natural Language Process** |

## LIST OF ATTACHMENTS

# CHAPTER 1: INTRODUCTION

## 1.1 Introduction

Nowadays, mobile device usage is increasing and it is expanded with new advanced technology every year. The continuous development of mobile hardware and technology has created a highly connected world. Mobile devices are used in many fields such as business, healthcare, education, and, Information Technology (IT). This transformation of mobile devices is attractive to the attacker to attempt to steal the information from the devices or system. Malware is malicious software that loaded onto mobile devices and causes damage then destroys it. The more malware attacks are introducing every year and it needs to be overcome before it widespread. This project focuses on mobile malware attack detection using a machine learning algorithm technique. This technique was chosen because it has good performance to track the malware and able to get high accuracy in classification.

## 1.2 Problem Statement

The mobile device becomes a place where intruders try their malicious code to fulfill their intention. There have many researchers proposed to prevent the mobile malware attack by introducing techniques that can protect the devices. Malware developers think it is easy to transfer attackers on mobile devices because mobile applications downloaded from Apple App Stores, Google Play, and websites usually have no time to analyze them (Gyamfi & Owusu, 2019). The current malware can do anything, their threat level becomes bigger and keeps growing. The variation of this malware and the exposed vulnerabilities require new and improved methods for detection.

There are plenty of attack situations in which an attacker can compromise a user's information by exploiting operating system's vulnerabilities. The growth of malware has led to different mobile antiviruses, firewalls, and encoding products, which have been released by various security vendors such as F-Secure, Kaspersky Lab, McAfee, and Symantec. The number of F-Secure mobile malware in 2008 was 401, while McAfee numbered 457. Mobile malware is malicious software that has certain mobile device targets. It first appeared for Symbian Operating System (OS) in 2004 and has now grown exponentially with smartphones in popularity (Gyamfi & Owusu, 2019). Figure 1.1 shows the mobile malware growth from 2010 up to 2018.

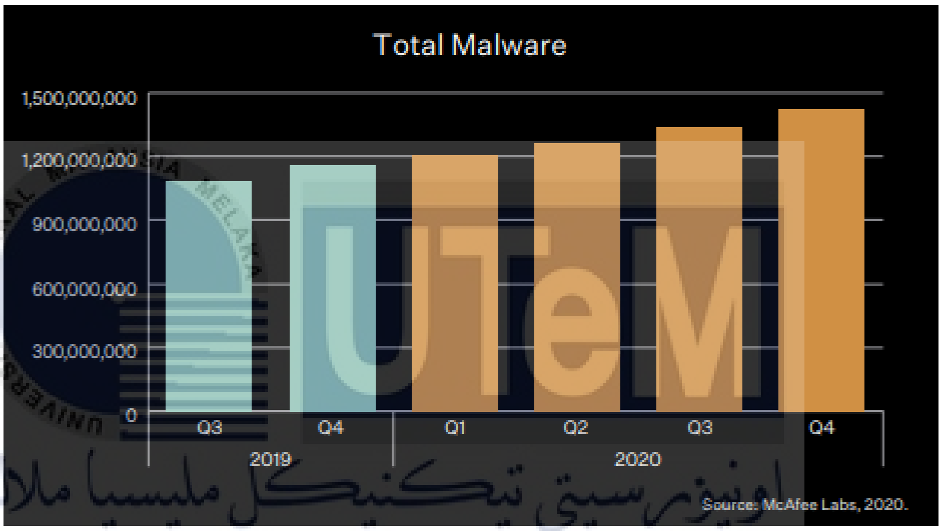


**Figure 1.1: Total Malware by Year (McAfee 2020)**

COVID-19 pandemic Movement Control Order (MCO) is another alarming issue with mobile malware attacks. According to McAfee (2020), McAfee Lab's average volume of malware threats was 588 per minute, up from 169 per-minute threats in the third quarter of 2020 (40%). The fourth quarter was 648 per minute threats, an increase of 60 per minute threats (10%). Mobile malware grew 118% by SMS Reg driven from Q3 to Q4 of 2020. The best solution to the problem is the detection of mobile malware. The problem was defined as in table 1.1.

**Table 1.1: Problem Statement**

| PS | Problem Statement |
|---|---|
| PS1 | An increasing number of users connect to the Internet has been driven by the tremendous growth in mobile and smart device usage, it is hard to discover effective and fast technique in differentiate between benign and malware. Another issue is to identify the suitable machine learning technique to use in this project to process the dataset. |

## 1.3 Project Question (PQ)

In this project, there are three Project Question (PQ) needed answers made out from the problem statement. Table 1.2 shows the summary of the project question based on the problem statement.

**Table 1.2: Summary of Project Question**

| PS | PQ | Project Question |
|---|---|---|
| PS1 | PQ1 | What is malware in mobile devices? |
| | PQ2 | How mobile malware classified? |
| | PQ3 | Is the machine learning technique give a result model accurately? |

## 1.4 Project Objective (PO)

Project Objective (PO) is a goal for the expected outcome of the research. To fulfill the goals of the project, the problem statement and project question must be assigned. Three objectives in this research to achieve and the relationship between PS, PQ, and PO for this project as shown in table 1.3.

**Table 1.3: Summary of Project Objective**

| PS | PQ | PO | Project Objective |
|---|---|---|---|
| | PQ1 | PO1 | To study the behavior of malware. |
| PS1 | PQ2, PQ3 | PO2 | To implement machine learning classifier to detect malware attack. |
| | PQ1, PQ2, PQ3 | PO3 | To test and verify the accuracy of machine learning to detect the malware. |

## 1.5 Project Scope

The scope of this project is related to the following criteria:

I. The dataset only focuses on mobile malware.

II. This project focus on machine learning method.

III. The effectiveness of the proposed model is measured using accuracy.

## 1.6 Project Contribution

The result of this project is depending on the project implementation to detect the mobile malware attacks and select effective features that will give high accuracy. The following is a brief description of this project's contribution:

I. Identification of different mobile malware attacks based on their taxonomy.

II. Propose a model that can detect malware attacks on the mobile device.

III. Propose to check the accuracy of mobile malware attack detection and test and validate the developed program.

## 1.7 Thesis Organization

This report divided to seven chapter namely Chapter 1: Introduction, Chapter 2: Literature Review, Chapter 3: Project Methodology, Chapter 4: Analysis and Design, Chapter 5: Implementation, Chapter 6: Testing and Validation and Chapter 7: Project Conclusion.

### 1.7.1 Chapter 1: Introduction

This chapter act as the guideline to identifying the important things prior to the project. It will explain what and how this project attempted to accomplish, as well as its relevance.

### 1.7.2 Chapter 2: Literature Review

This chapter consists the related studies and early task from previous researcher about this project and critical analysis of this chapter.

### 1.7.3 Chapter 3: Project Methodology

Chapter 3 focuses on planning works that must answer the objectives of this project that has been mention previously. This chapter includes the procedures throughout this project.

### 1.7.4 Chapter 4: Analysis and Design

This section is related to the procedure that correlate with the experiment of this project. The experiment needs to be analyzed the effectiveness by referring to the literature review and design the project clearly to reach the goals.

### 1.7.5 Chapter 5: Implementation

Chapter 5 is the main part that the procedure from the previous chapter needs to be carried out. The experiment will be processed in this section until has the result and it will be gathered the result to be compared with another method.

### 1.7.6 Chapter 6: Discussion

This chapter explains the result and analyzed them where the findings are answers the objectives.

### 1.7.7 Chapter 7: Project Conclusion

This chapter will summarize the project, state the contribution, and highlight the constraints that were encountered throughout the project. This chapter will also describe how to improve the project in the future.

## 1.8 Conclusion

This project is carried on to get more information in mobile malware attack detection to differentiate between benign or malware. This project also should be able to distinguish the benign and malware attack by using the machine learning method to develop a model. Then the model needs to be tested their accuracy by using different techniques. The next chapter discusses the literature review based on mobile malware attacks and machine learning.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 Introduction

This chapter will discuss and review regarding the mobile malware attack and how to identify that attack, as in shown in Figure 2.1. At the end of this chapter, a better understanding will be gained on mobile malware detection using machine learning method. This chapter contains description about the definition, categories, classification, approaches and techniques which have been used by previous researchers to provide literature review. The literature review will be divided into several parts to illustrate mobile malware, machine learning, feature selection, and the method of division and classification used by researchers during the literature review.

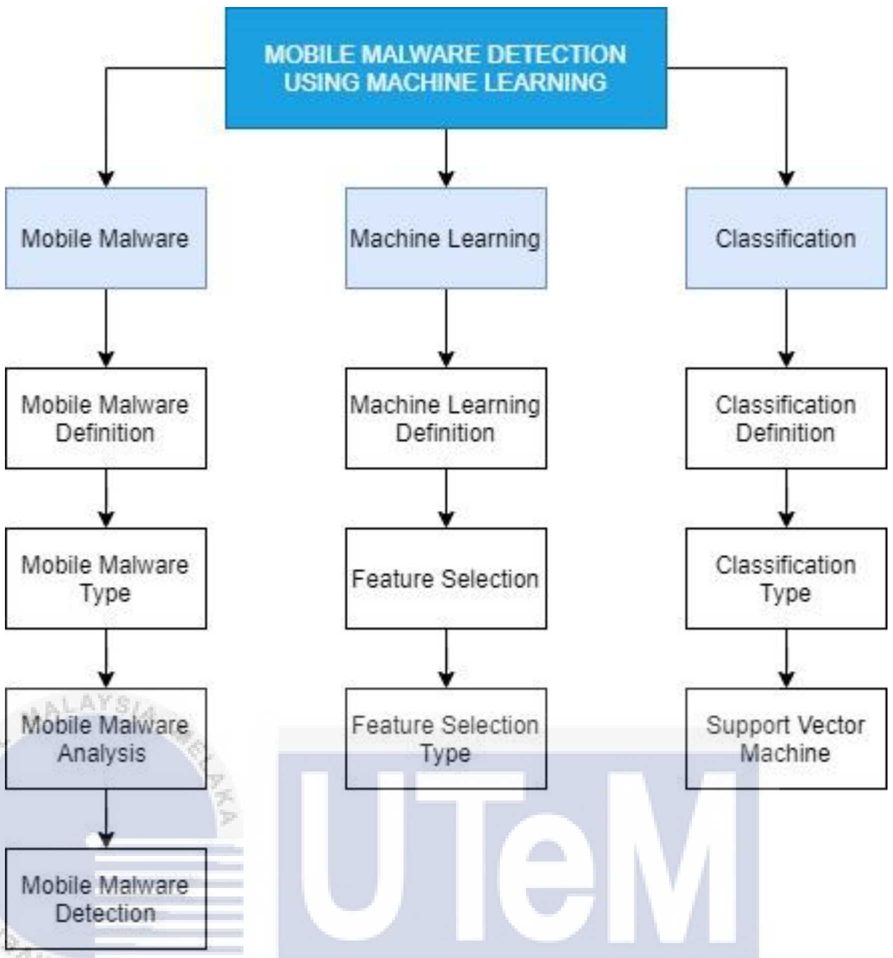


**Figure 2.1: Literature Review's Structure**

## 2.2 Mobile Malware

### 2.2.1 Mobile Malware Definition

The growth of technology when everything is connected through cyberspace will raise the risk to security. Nowadays, mobile devices have popularity in low-cost technologies that provide social media platforms, cloud computing, and social networks that can get any threat, misconfigured devices, and potentially vulnerable to harm. Mobile malware can be defined as malicious software that targets the operating system on mobile devices to access and steal any sensitive data. Suppose malware is identified as soon as possible. In that case, the safety and protection of data, corporate information, identification, and mobile phone resources used in IoT networks can be jeopardized. This attack could cause loss of data, damage to the smartphones, and can no longer be used.

The mobile malware software executes itself when using a mobile computer to access the internet and then performs functions without the user's knowledge or approval. They are spreading in a variety of ways, including the internet via a mobile browser, downloads, and installation via application messaging functions. Malware for mobile devices can be divided into many categories (Gyamfi & Owusu, 2019). According to research (Sharmeen et al., 2018), The resource usage of mobile devices' operating systems is always a concern. Because of the limited resources available, malware detection is a complex problem to solve on the mobile platform. Researchers face a challenge in detecting malicious activities in mobile apps using limited resources in a short amount of time.

### 2.2.2 Mobile Malware Type

Every year the among of malware evolution is increasing continuously and difficult to handle. To attacks mobile devices, malicious hackers employ a variety of methods. Many mobile malware varieties can be spreading themselves and corrupt the system of the device. The standard malware lists are including Trojans, worms, botnets, spyware and ransomware. The mentioned below is the latest and most popular mobile malware, according to Kaspersky.

Banking malware is a form of malicious software that steals data from financial institutions. The original developers or those using leaked source code have continued to release updated models of banking malware families. Many of them are Trojans that penetrate computers and then deploy, gathering bank login and password information before sending it back to a command and control (C&C) server. Once installed, the Banker Malware will gain access to mobile files and systems, allowing attackers to carry out unauthorized transactions, steal clients' identities, and transfer funds from clients' accounts.

Mobile Ransomware occurs by encrypting sensitive user data, including documents, images, and videos, ransomware locks out the data and requests a ransom to the malware's creators. Users are often scammed into installing mobile ransomware through social networking schemes, believing they are downloading harmless material or essential apps. If the ransom is not paid promptly, typically, in Bitcoin, all files are removed or locked, rendering the user permanently unavailable.

Mobile spyware is installed as software on a computer, tracks device activity, monitors locations, and steals sensitive data, including usernames and passwords for email accounts and e-commerce sites. Spyware is often bundled with other harmless applications and secretly gathers data in the background. Users may not realize spyware is present until their device's functionality deteriorates or the user run an anti-malware scanner on a tablet or phone.

MMS malware developers are now searching for new ways to use text-based communication to spread malware. It became possible for attackers to send a malware-infected text message to any cell phone number. The malware could still install even if users will not open or acknowledge the text, giving hackers root access to the user phone. The issue was quickly fixed, but it provided evidence of text-based infections.

Mobile adware has come a long way since irritating pop-ups and data collection is what it was known for. According to ZDNet, several adware creators have now developed malvertising code that can corrupt and root your computer, forcing it to download new adware variants and enabling intruders to steal sensitive information.

SMS trojan, malicious hackers compromise mobile devices by preying on consumers' features like the most, such as text messages. SMS trojans cause financial chaos by sending SMS messages to premium-rate numbers worldwide, racking up phone bills for their victims. In 2015, a banking trojan infected some Android users, allowing cybercriminals to intercept text messages containing financial data and then send a copy version of the text message via email, giving them all the information, they needed to hack into financial accounts.

### 2.2.3 Mobile Malware Analysis Technique

The static analysis looks at a code without running it. While it can expose all possible execution routes, it has many drawbacks. Traditional studies and signature methods such as Windows whole-file, segment, and code hashing are incompatible with Android due to alternative code compilers. However, (Tam et al., 2017) the research found that all static techniques are vulnerable to obfuscations like encryption preventing or restricting access to the code.

Dynamic analysis, the apps are run in a virtual environment or on an emulator to understand dynamic behavior such as permission use, device call tracing, taint tracking, CPU usage, battery analysis, and RAM consumption during dynamic analysis (Kalpana & Karthikeyan, 2018). Taint monitoring entails analyzing data flow from sensitive sources such as cameras, GPS, and microphones. It avoids the issues caused by obfuscation techniques and polymorphic malware, unlike static analysis.

According to (Galib & Mainul Hossain, 2019), For maximum efficiency, the hybrid analysis combines both static and dynamic functions. Its first goal is to extract the static and dynamic features of Android apps. Following that, the static and dynamic features obtained are combined to create a detection model. Finally, a detection model is developed using machine learning techniques to identify Android malware based on static and dynamic features.

### 2.2.4 Mobile Malware Detection Technique

Malware detection techniques are used to determine the malware and prevent it from infecting the operating system, preventing data loss and system failure. Signature-based detection and anomaly-based detection are two types of detection as shown in Table 2.1.

**Table 2.1: Mobile Malware Detection Techniques**

| Technique | Description |
|---|---|
| Signature-based detection | Gathers patterns and signatures from malicious code and compares them to suspicious code to determine if it is malicious or benign. |
| Anomaly-based detection | Observing a device's normal behaviour for a period of time and then using the metrics of that normal model as a reference vector to deviant behaviour |

Static, dynamic, or hybrid methods may be used for each detection technique. An anomaly-based or signature-based technique's primary method or evaluation is dictated by how the technique gathers information to detect malware.

A static method, in general, can detect malware before the software under the inspection category executes it. A dynamic method, on the other hand, detects malicious activity during or after programme execution. The final method is hybrid, which combines static and dynamic techniques to detect malware. Figure 2.1 shows the relationship between the methods and their approaches.

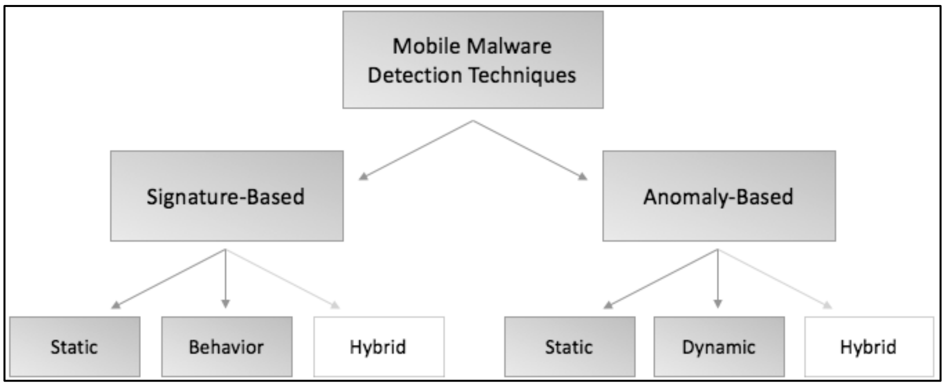


**Figure 2.2: Mobile Malware Detection Classification (Kouliaridis et al., 2020)**

## 2.3 Machine Learning

### 2.3.1 Machine Learning Definition

Machine Learning (ML) is a part of artificial intelligence (AI) and computer science that uses data and algorithms to mimic humans' learning, intending to continually improve accuracy. The goal of machine learning is to gain knowledge from data. There have been many experiments on how to teach computers to learn on their own. To solve this problem, many mathematicians and programmers use a variety of approaches (Dey, 2016).

There are three major type in machine learning including supervised learning, unsupervised learning and reinforcement learning. The used of these types of machine learning is depending to the algorithm and the objectives as stated in Table 2.2 below.

**Table 2.2: Type of Learning**

| **Type of Learning** | **Description** |
|---|---|
| Supervised Learning | Algorithms that require external aid are known as supervised machine learning algorithms. The training and testing datasets are separated from the input dataset. There is an output variable in the training dataset that needs to be predicted or classified. |
| Unsupervised Learning | Only a few features are learned from the data by unsupervised learning methods. When new data is presented, it recognises the data's class using previously learnt features. |

| | |
|---|---|
| Reinforcement | Reinforcement learning is a method of learning that involves making decisions about which actions to perform to improve the outcome. Until a circumstance is presented to the learner, it has no idea what actions to take. |

### 2.3.2 Feature Selection

Definition of feature selection is by deleting irrelevant, redundant, or noisy characteristics. Feature selection as a dimensionality reduction strategy seeks to choose a small subset of the useful features from the original ones. Feature selection frequently leads to improved learning performance, such as increased learning accuracy, lower computing cost, and more interpretable models (Suhang Wang et al., 2016). A set of features determines the hypothesis of the prediction models. The number of features and the hypothesis space are directly related, meaning that as the number of characteristics grows, so does the hypothesis space (Venkatesh & Anuradha, 2019).

In general, irrelevant features are incapable of distinguishing samples from various classes (supervised) or clusters (unsupervised). Removing non-essential characteristics has little effect on learning. In reality, removing irrelevant features may aid in the development of a better model, as unnecessary features might mislead the learning system and lead to inefficient memory and computation.

#### 2.3.2.1 Type of Feature Selection

Several types of strategies will be used in variable selection in feature selection. Feature selection will be made for lower dimension datasets to make it easier to discover the best subset without changing the data into a new set. Aside from that, variable selection will be broken down into many methods, including filters, embedded, and wrapper method. The description of these three types is as in Table 2.3.

**Table 2.3 Feature Selection Summary**

| **Model** | **Description** |
|---|---|
| Filter | Features are chosen based on the characteristic of data without depending on the employed learning algorithm. The modelling algorithms can only employ the best features after they have been discovered. |
| Wrapper | Wrappers evaluate feature subsets based on the quality of a modelling algorithm's performance, which is used as a black box evaluator. |
| Embedded | Embedded techniques choose features during the execution of the modelling process. As a result, these methods are either standard or enhanced functionality in the algorithm. |

In filter model, individual features can be ranked, or entire feature subsets might be evaluated using filter algorithms. Not all filter characteristics apply to all types of data mining activities. As a result, the filters are divided into categories based on the tasks, such as classification, regression, or clustering. Multivariate feature filters assess a whole feature subset, whereas univariate feature filters analyse and usually rank a single feature. The search strategy influences the generation of feature subsets for multivariate filters.

Filter approaches is divided into three which are Information Gain (IG), Chi-Square (CS) and Correlation method. Information gain is a symmetrical measure of two-variable dependency. The value of a feature is assessed by how much the class's entropy decreases when the related features are analysed individually. The disadvantage of IG is that it prefers characteristics with higher values, even if they are not necessarily more informative. The formula below is about IG, after viewing X, the information obtained about Y is identical to the knowledge received about X after observing Y (Anukrishna & Paul, 2017)

$$IG (X; Y) = H (X) - H (X |Y)$$

In statistics, the Chi-Square (CS) test is to determine if two events are independent. We can extract the observed count O and expected count E from the data of two variables. The Chi-Square test determined how far predicted count E and observed count O differ. Simply put, the higher the Chi-Square value, the more dependent on the response the feature is, and it can be chosen for model training. While the correlation-based feature selection approach evaluates feature subsets by picking feature subsets that comprise features that are substantially correlated with the classification but not with each other.

For classification tasks, a wrapper evaluates subsets using a classifier such as Naive Bayes or SVM. For clustering tasks, a wrapper will evaluate subsets based on the performance of a clustering algorithm such as K-means. Each subset is repeated for the evaluation same with filters, and the subset generation is based on the search strategy. Because wrappers are dependent on the resource demands of the modelling algorithm, they are much slower than filters at finding sufficiently good subsets. The feature subsets are also biased in favour of the modelling algorithm used to evaluate them. On the other hand, wrappers have been observationally proven to produce better performing subsets than filters because the subsets are tested using an actual modelling algorithm.

Embedded approaches include decision tree algorithms such as CART, C4.5, and random forest, as well as other algorithms. Some embedded approaches execute feature weighting using regularisation models with goal functions that reduce fitting errors while forcing feature coefficients to be minimal or exact zero in the meantime.

## 2.4 Classification

### 2.4.1 Classification Definition

Classification is the task of determining whether an object belongs to a specific category based on a previously acquired model in machine learning or statistics. This model is learned statistically from a collection of training data with predefined categorization. Although classification is a well-known machine learning methodology, it has flaws, such as missing data handling. Both the training and classification processes might be affected by missing values in data collection (Soofi & Awan, 2017).

### 2.4.2 Type of Classification

Unsupervised machine learning is drawn inferences from datasets that contain input data but no labelled answers. The goal of supervised machine learning approaches is to discover the relationship between input and target attributes. There are many classification techniques in supervised machine learning as in Figure 2.4 below.

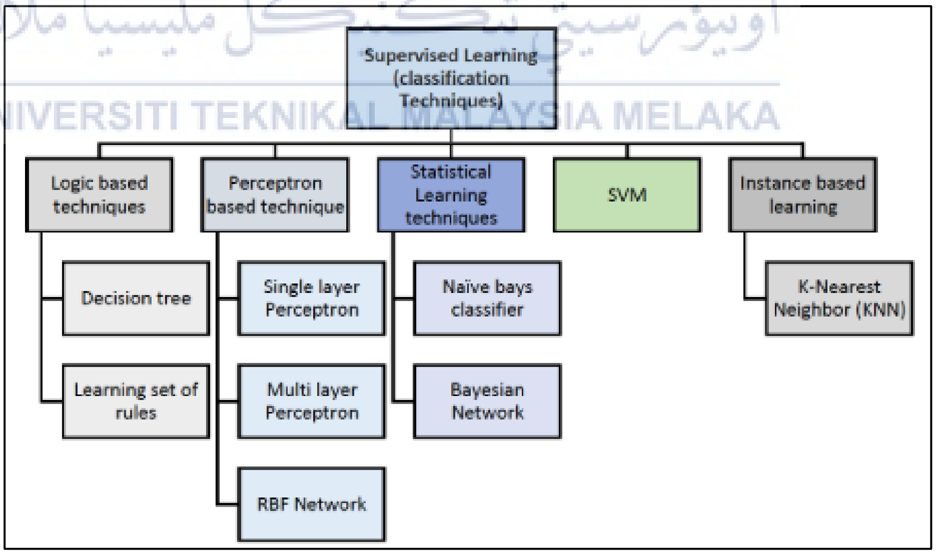


**Figure 2.3: Classification Techniques in Supervised Learning (Soofi & Awan, 2017).**

Decision trees are trees that group qualities by ordering them according to their values. The decision tree is mainly used for classification. Nodes and branches make up each tree. Each branch indicates a value that the node can take, and each node represents attributes in a group that needs to be categorised as shown Figure 2.3. The main goal of a decision tree is to create a model that determines the value of a necessary variable using a variety of input variables. All decision tree algorithms are usually built in two stages. The first phase is a tree growth, in which the training set is partitioned recursively based on local optimum criteria until the majority of the records in the partition have the same class label. The second phase of tree pruning involves reducing the size of the tree to make it easier to understand.

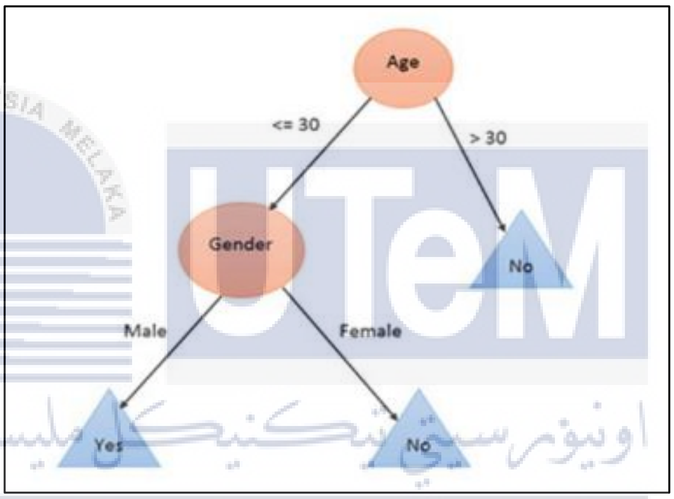


**Figure 2.4: Decision Tree**

**(Dey, 2016)**

Bayesian Networks made up of directed acyclic graphs with only one parent representing the unobserved node and multiple children corresponding to observed nodes, with a strong assumption of child node independence in the context of their parent (F.Y et al., 2017). Disadvantages of Bayesian network classifiers is that continuous attributes must generally be discretized. Converting a continuous attribute to a discrete attribute resulted in classification difficulties. According to (Soofi & Awan, 2017) research to overcome this problem is by using Gaussian kernel function in Bayesian Network classifiers.

Smoothness qualities such as slight changes in the Bayesian network model do not affect the system's operation. Flexible applicability such as identical Bayesian Network models may be utilized to resolve both regression and classification challenges. Finally, handling missing data are some of the advantages of Bayesian networks.

Multilayer perceptron, instead of addressing a non-convex, unconstrained minimization problem as in traditional neural network training, the network weights are discovered by solving a quadratic programming problem with linear constraints. The perceptron algorithm is used to learn from a batch of training examples by repeatedly running it over the training set until it discovers a correct prediction vector across the board. The labels on the test set are then predicted using this prediction rule.

In K-Nearest Neighbour (KNN), the value of k, which defines how many nearest neighbours must be examined to describe the class of a sample data point, is used to measure nearest neighbour. One of the key advantages of the KNN methodology is that it works well with large amounts of training data and is resistant to noise. The space demand and classification time of nearest neighbour-based classifiers are identified as two major drawbacks. Based on (Soofi & Awan, 2017), K-Nearest Neighbour Mean Classifier (k-NNMC) was introduced to overcome the space issues.

Last but not least, classification technique in supervised learning is Support Vector Machine (SVM). It is primarily used for classification. SVM is based on the principle of calculating margins. The data points nearest to the decision surface are called support vectors. It uses a hyperplane to classify data vectors in an enormous dimensional environment. The simplest or most basic type of SVM is the maximum margin classifier, which aids in determining the most fundamental classification problem of linear separable training data with binary classification. Support Vector Machine's primary advantage is its ability to deal with a wide range of classification problems, including high dimensional and not linearly separable. One of the most significant disadvantages of SVM is that it needs the proper setting of some essential parameters to achieve high classification results (Soofi & Awan, 2017).

One of classification techniques in unsupervised learning algorithms is K-means that is the simplest one to solve the clustering issues. The approach uses a simple and straightforward method to classify a given data set using a predetermined number of clusters assuming k clusters. When labelled data is not available, the K-Means algorithm is used. A general strategy for converting sloppy guesses into highly accurate prediction rules.

## 2.5 Two-Layer Stacking SVM Classifier

Two-Layer Stacking SVM Classifier is one of the deep learning machine learning approaches. According to (Abdullah et al., 2009), they have difficulties with their classifier approach, which increases the size of the descriptor and their feature vector becomes larger that can affect the SVM training. They use the two-layer stacking method to solve these problems that can help reduce feature vector and weighting scheme. In that research, the combination method of the two-layer spatial stacking with the outputs spatial pyramid approach of the different classifiers was used.

The algorithm will train a set of SVM classifiers first on each level histogram. Each classifier calculates the class probability values or class predictions in the posterior. The probabilities are used instead of the feature vector at the final classifier because it has predicted classes information. The outputs of these classifiers will enter another SVM classifier and get the final result. Figure 2.5 shows the illustration of two-layer stacking method.

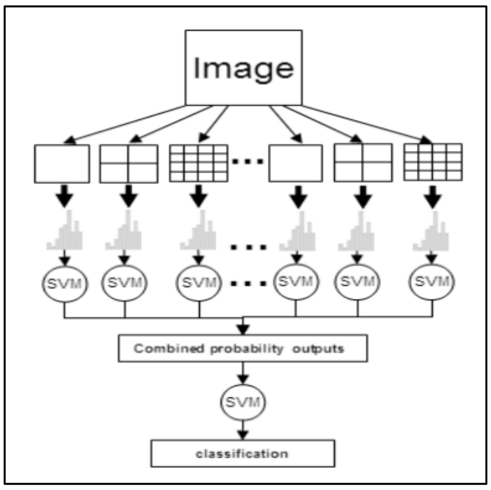


**Figure 2.5: Two-Layer Stacking Spatial Pyramid Classifier**

## 2.6 Critical View

### 2.6.1 Previous Research on Mobile Malware

In this era of technology, mobile phones have become a popular device with the ability to connect through the internet. The fast transformation of mobile devices attracts the intruder to send malware to the devices. Furthermore, as researchers develop new detection systems to prevent mobile malware, recent studies indicate that mobile malware is increasingly exploiting victims to create profit (Kouliaridis et al., 2020). In this paper discusses an overview of the different approaches to mobile malware detection.

Since mobile devices are linked to third-party applications, a slew of security and privacy issues arise. Current mobile malware detection and analysis tools, on the other hand, are still inefficient, ineffectual, and insufficient. After a mobile app is installed or executed, or when a mobile network is linked, many intrusion or attacks may occur (Yan & Yan, 2018). In this paper will discuss about survey on dynamic mobile malware detection and provide a variety of performance evaluation metrics and criteria for detecting mobile malware.

According to the research, Android is the most extensively used platform among all platforms. With the increasing prevalence of employing these mobile platforms in sensitive applications, malware targeting mobile devices has become a threat. Malware is any code that is introduced, updated, or removed from an application with the intent of causing harm to the system's intended function (Malhotra & Bajaj, 2016). This paper discusses a various malware detection technique on mobile platform. There two categories of malware detection techniques include anomaly-based detection and signature-based detection (Misuses-based).

**Table 2.4: Survey on Mobile Malware Detection**

| **Author** | **Title** | **Result/Description** |
|---|---|---|
| (Kouliaridis et al., 2020) | A Survey on Mobile Malware Detection Techniques | This paper discussed a mobile malware detection literature 2011 to 2018 with the result. For signature-based detection, the result achieved 91.6% detection rate. In the anomaly-based detection, the highest accuracy rate achieved is 99.4% with 16.1% false positive rate. They test their classification using Naive Bayes, K-nearest Neighbour, Random Forest, and Support Vector Machine algorithms |
| (Yan & Yan, 2018) | A survey on dynamic mobile malware detection | This paper focuses on dynamic malware detection proposed since 2013. They contrasted previous research in terms of technological categories, classification techniques, detection features, and target mobile operating systems. The locations of detection analysis, real-time detection assistance, privacy preservation assistance, risks that can be exposed and overcome, |

| | | |
|---|---|---|
| | | detection performance assessment measures, and performance test results were also taken into consideration. |
| (Malhotra & Bajaj, 2016) | A Survey on Various Malware Detection Techniques on Mobile Platform | This paper found that detection framework called Malicious Sequential Pattern based Malware Detection (MSPMD) combine All-Nearest-Neighbour (ANN) classifier achieved 96.17% detection rate, 6.13% false positive rate and 95.25% accuracy compare to another framework. |

### 2.6.2 Previous Research using Machine Learning

Android is an operating system that is rapidly increasing in popularity and allows users to download and install applications from an unauthorized marketplace. According to a study (Arshad et al., 2018), this is attractive to malware developers to create third-party malicious software and attack the Android operation system's devices. So, in this study, they proposed a novel 3-level hybrid malware detection model called SAMADroid. The 3-level model is including static and dynamic analysis, local and remote host and the last one is machine learning intelligence. Dataset in this study is from Drebin's dataset. To achieve high accuracy, they use Drebin features for level 1 while at level 2 they performed a detailed static analysis on host of remote and feature vectors are used in level 3. In first experiment, Linear Support Vector Machine (SVM) classifier was used and repeat the experiment for 10 times. In second experiment, they performed Decision Tree, Naïve Bayes and Random Forest classifier and also repeat for 10 times to get high accuracy.

Mobile malware can be insert by itself without user's knowledge and this could lead to leakage of sensitive data like business information, user login credentials and account information (Anuar, Mas'ud, Bahaman, & Ariff, 2020). In this paper, to mitigate this attack by performed opcode sequence to machine learning classifier to

Android malware detection. 500 malicious.apk were from ArgusLab and 500 benign software from Google Play Store. There are three phases involved which are data collection, feature selection and machine learning classifier. Classification algorithms that applied in this project are Naïve Bayes, Support Vector Machine, Random Forest and Decision Tree (J48). The experiment was repeated 10 cycles then the series of performance metrics are measure to shows the efficiency of the classifier.

Mobile malware is increasing growing and become harder to resolve. A popular thread that needs to face now is network-based ransomware. Malicious apps use various techniques to get through the Android operating system's or anti-virus software's detection measures. (Shanshan Wang et al., 2019) research used a device-level lightweight malware identification and classification framework. They performed a network traffic analysis to expose malware malicious trace by implement traffic mirroring. This paper combines two methods, including network traffic analysis with a machine learning algorithm called Decision Tree (C4.5), to achieve high accuracy. By altering the number of samples in each family, the performance of malware detection on unknown families is assessed.

**Table 2.5: Previous Research using Machine Learning's Literature**

| **Author** | **Title** | **Result/Description** | **Dataset** |
|---|---|---|---|
| (Arshad et al., 2018) | SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System | Based on experimental result from this paper, the SVM classifier achieved 98.97% accuracy and 0.5% False Positive Rate (FPR). The highest accuracy gained is 99.07% and 0.07% FPR by using Random Forest. Decision Tree gained 98.56% accuracy and 0.5% FPR while Naïve Bayes is 91.6% accuracy and 7.8% FPR. | Drebin's dataset |

| | | | |
|---|---|---|---|
| (Anuar, Mas'ud, Bahaman, & Ariff, 2020) | Analysis of Machine Learning Classifier in Android Malware Detection through Opcode | Based on this project, shows the highest accuracy achieved is 0.954 by using Support Vector Machine (SVM). While Naïve Bayes (NB) and Decision Tree (J48) have same accuracy rate which is 0.899. For Random Forest (RT) classifier gained 0.893 accuracy rate. | Android Malware Dataset by ArgusLab |
| (Shanshan Wang et al., 2019) | A mobile malware detection method using behaviour features in network traffic | Based on this research result, when they combine two method which are network traffic analysis and Decision Tree (C4.5) algorithm, accuracy gained is 97.89%. | Drebin's dataset |

### 2.6.3 Previous Research using Feature Selection

For differentiating between benign and malicious traffic, network traffic-based malware detection techniques rely on chosen traffic features. (Shanshan Wang et al., 2017) proposed two natural language process (NLP) method which are word segmentation and N-gram model. To extract important features, TextDroid uses an automatic feature selection technique which is Chi Square based on N-gram sequences. Next, they train the extracted features to Support Vector Machine (SVM) classifier and achieved high detection rate.

(Mas'ud et al., 2017) stated the main goals of feature selection stage is to enhance the performance of mining and also to improving the accuracy by reduce the unimportant features in dataset. In this experiment, Chi Square (CHI), Correlation-based Feature Selection (CFS), Information Gain (IG) and ReliefF (RF) were used in filter method. For wrapper method, they choose Linear Support Vector Machine classifier (WR). For all of the features created by the feature selection algorithm, the evaluation demonstrates an improvement in accuracy.

(Singh & Hofmann, 2018) performed extracted system call by malicious application and normal during execution. Feature selection and ranking techniques were employed to improve classifier performance. There are seven machine learning were used which are Support Vector Machine, Deep Learning, Decision Trees, Neural Network, K-nearest Neighbours, Random Forest, and Gradient Boosted Trees. In the experiment, this project using machine learning classifier first then they applied the feature selection to the classifier. Feature selection and ranking this project executed were Information Gain, Chi Square and feature selection using correlation.

**Table 2.6: Previous Research using Feature Selection's Literature**

| Author | Title | Result/Description | Dataset |
|---|---|---|---|
| (Shanshan Wang et al., 2017) | TextDroid: Semantics-based Detection of Mobile Malware Using Network Flows | This paper used Chi Square to produce a meaningful feature with N-gram model. The result gained is 96.36% detection rate in test set using SVM algorithm but in real environment they gained 76.99% detection of malicious application | Malicious network traffic produced by malicious application, and benign network traffic produced by benign application |
| (Mas'ud et al., 2017) | A Comparative Study on Feature Selection | Based on the result in this paper, feature ranking method used alongside Chi Square and | MalGenome Project and Google Play. |

| | | | |
|---|---|---|---|
| | Method for N-gram Mobile Malware Detection | Information Gain obtained 96.0% accuracy while ReliefF (RF) gained 95.0% accuracy. The highest accuracy achieved is Linear SVM with BestFirst method which is 99.0% | |
| (Singh & Hofmann, 2018) | Dynamic behavior analysis of android applications for malware detection | The result show in this paper, the highest classifier after feature selection using Information Gain is Gradient Boosting Trees with 98.38% accuracy. For Chi Square with Gradient Boosting Trees also same 98.38% and Correlation is 99.19%. But the three features selection achieved highest and equal accuracy with SVM which is 99.54% for recall. | Google Play Store and cantiago project |

## 2.7 Conclusion

In conclusion, this chapter give a better understanding on the goals of project about mobile malware, machine learning, feature selection and critical view. It is possible to do so by studying previous literature reviews written by other researchers. In the critical review topic stated the comparison from previous research related to mobile malware detection, machine learning approach and Support Vector Machine method. From the comparison, it can use to develop a proper framework of research in mobile malware detection. Thus, the proposed methodology will further be explained in the next chapter which is chapter 3.

# CHAPTER 3: PROJECT METHODOLOGY

## 3.1 Introduction

This chapter discusses the methodology this project used and ensures this project complete within the time given. The methodology in this project is chosen by referring to the previous chapter, which is a literature review that explains the previous research. This chapter creates a framework based on the domain explain in the previous chapter. This framework to make sure this research follows the procedures and steps throughout this research. A flowchart develops to highlight the process in this project and a milestone and a Gantt chart to ensures this project complete in the given timelines.

## 3.2 Methodology

The methodology is one of the ways or processes involved in solving a problem. This phase is essential to ensure that this project completely follows the flow proposed. In this research, a framework is select according to the best and suitable machine learning approach to mobile malware detection. There are six phases to implemented and make sure this project process smooth. The six phases involved are previous research, information gathering, defining scope, design and implementation, testing and evaluating the model, and documentation.

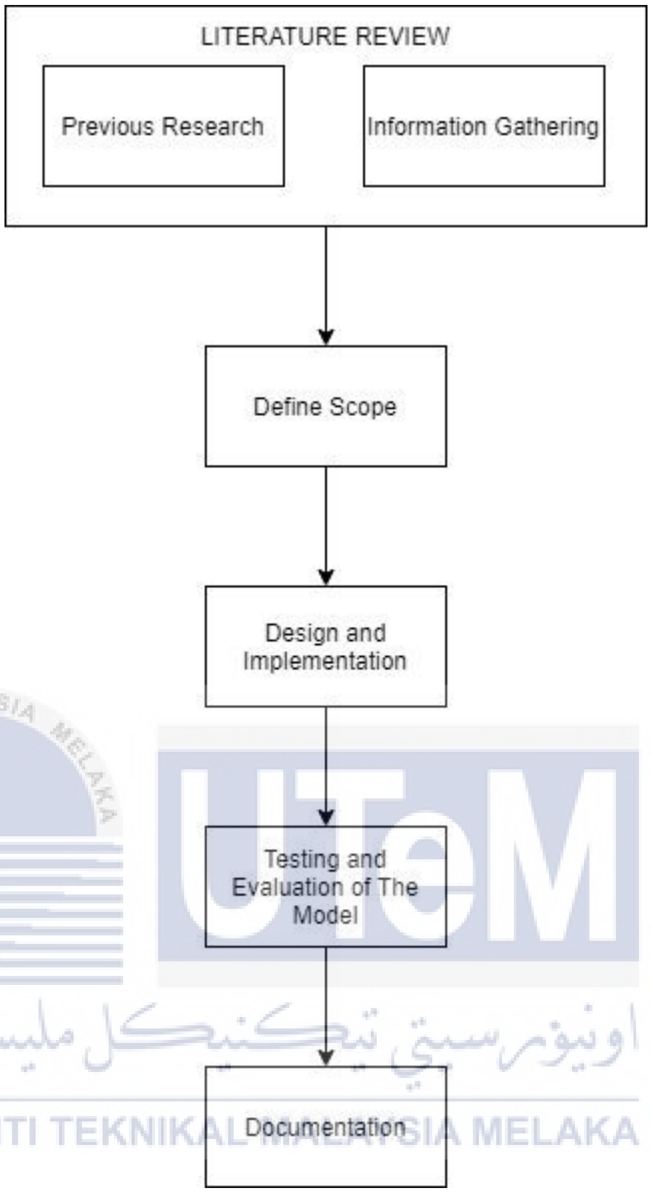


**Figure 3.1: Flowchart for the System Framework**

### 3.2.1 Previous Research

This phase gives a clear understanding of how to operate this project according to the previous research. It is crucial to ensure all the specific requirements are defined because a domain is required before this project runs. The domain for this research is mobile malware detection, machine learning, and classification. Previous research will reveal how the proposed theoretical frameworks function in their respective fields. The details of these things have been discussed in this phase.

### 3.2.2 Information Gathering

Information Gathering gives a strong understanding of the research matter in previous research. All information obtained during this phase is the last chapter explaining the analysis and method using the machine learning model. This research will develop a model using a Support Vector Machine (SVM) algorithm. Referring to information gathering in the previous study will strengthen the chosen algorithms and techniques used to run experiments in this project.

### 3.2.3 Define Scope

The scope is restricted where the research run to analyse the data about mobile malware issues. This project will focus on Support Vector Machine classification to develop a model to detect mobile malware detection. The precision accuracy and confusion matrix between the created model and the benchmarks were examined.

### 3.2.4 Design and Implementation

This phase will detail the design and implementation of the proposed models from the previous phase study. Firstly, the dataset was obtained from previous research by Mohd Zaki Mas'ud. Then, the dataset will separate into two, which are the test set and train set, and go through a feature selection process. The features selection used is Information Gain and Chi-Square to remove irrelevant and redundancy features. The new features set from the feature selection used in classification using Support Vector Machine (SVM) to develop a new model detection.

### 3.2.5 Testing and Evaluation of Model

The testing and evaluation phase is used to see if the developed model meets the specified requirements. In this scenario, the accuracy of the features created at the end of the process is compared. However, the expected that analysis using SVM would improve the accuracy of existing research material. As a result, this research aims to put this hypothesis to the test in several different experiments. Comprehensive design analysis will guide future work on the research's development.

### 3.2.6 Documentation

The documentation phase is to organize the result properly and more structured. Each experiment needs to write the procedures appropriately involved and the results gained to proper documentation because this is proof for every activity. To achieve this project's objectives requires a lot of procedures. Without proper documentation, this project will mess. All relevant data is first discovered and reported in the respective parts.

## 3.3 Project Schedule and Milestones

### 3.3.1 Project Flowchart

A flowchart is used to create a systematic summary of the tasks and their relationships. This procedure defines the essential resources that must be mapped to their appropriate activities to avoid delays or potential limits. Figure 3.2 shows the flow diagram of the general phases of the project.

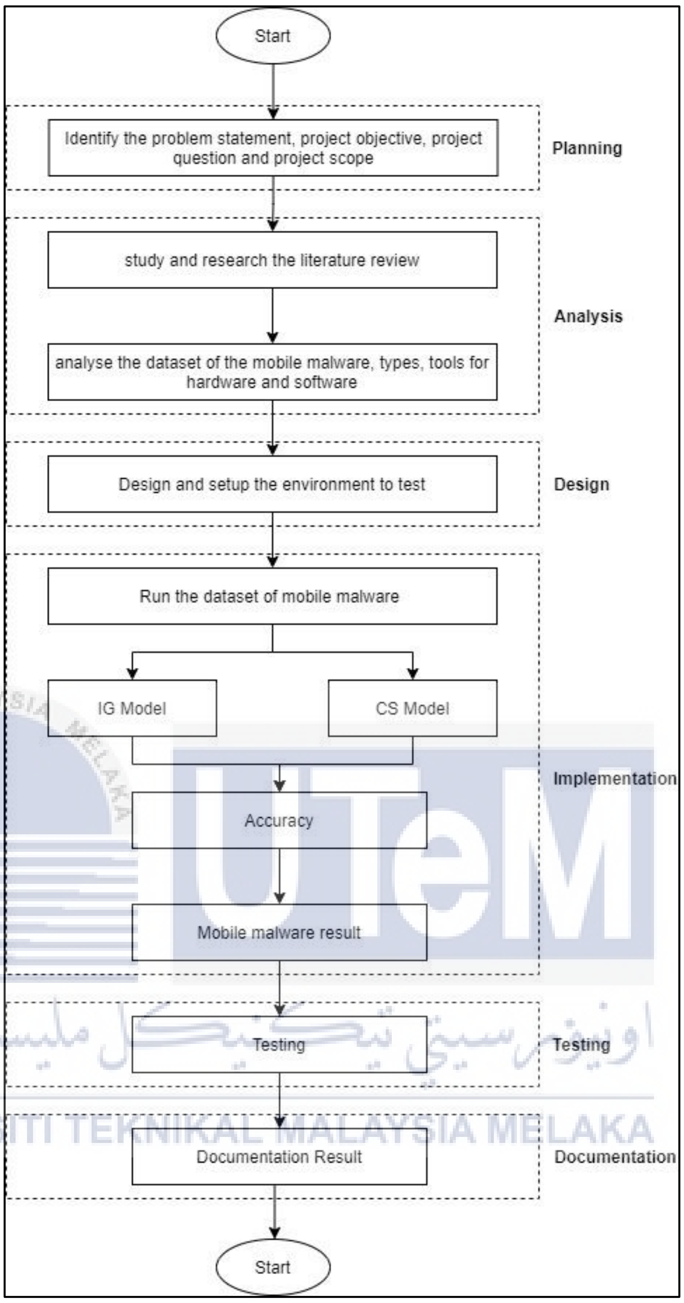

Start
Identify the problem statement, project objective, project question and project scope
Planning
study and research the literature review
analyse the dataset of the mobile malware, types, tools for hardware and software
Analysis
Design and setup the environment to test
Design
Run the dataset of mobile malware
IG Model
CS Model
Accuracy
Mobile malware result
Implementation
Testing
Testing
Documentation Result
Documentation
Start

**Figure 3.2: Project Flowchart**

### 3.3.2 Project Milestones

A project milestone is one tool management that keeps track of the upcoming events and objectives throughout specific times. Add the significant value in milestones can help detect the project is follow the schedule or not. Figure 3.1 below shows the milestones of this research.

**Table 3.1: Project Milestones**

| Week | Phase | Action | Deliverables |
|---|---|---|---|
| 1-4 | Planning | (15/3/2021 – 21/3/2021) Identify title, problem statement and scope. | Project Proposal |
| | | (22/3/2021 – 28/3/2021) Study and research the literature review. Prepare and submit proposal to supervisor. | |
| | | (29/3/2021 – 4/4/2021) Proposal accepted. | |
| | | (5/4/2021 -11/4/2021) Chapter 1: introduction, problem statement, project question, project contribution. | Chapter 1: Introduction |
| 5-8 | Analysis | (12/4/2021 – 18/4/2021) Research on related work and previous research about mobile malware. | Chapter 2: Literature Review |
| | | (19/4/2021 – 25/4/2021) Write the critical review about model used by previous researchers | |
| | | (26/4/2021 – 2/5/2021) Study methodology on previous research. | Chapter 3: Project Methodology |
| | | (3/5/2021 – 9/5/2021) Information collection and analysis. | Chapter 4: Analysis and Design |

| | | | |
|---|---|---|---|
| 9 | (10/6/2021 – 16/5/2021) - MID SEMESTER BREAK | | |
| 10-14 | Design | (17/5/2021 – 30/5/2021) Design the project ang choose the tools for implementation. | Chapter 4: Analysis and Design |
| | | (31/5/2021 – 13/6/2021) Design the environment for implementation. | |
| | | (14/6/2021 – 20/6/2021) Progress report on Chapter 4 | |
| 15 | (21/6/2021 – 27/6/2021) – Final Presentation | | |
| **Short Semester** | | | |
| 1-4 | Implementation | (19/7/2021 – 15/8/2021) Code the logic based on the designed algorithm and test using the dataset. | Chapter 5: Implementation |
| 5 | Discussion | (16/8/2021 – 22/8/2021) Monitor the performance of newly proposed method ang tuning. | Chapter 6: Discussion |
| 6 | Project Conclusion | (23/8/2021 – 29/8/2021) Highlight summary of the project, constraints, and future works | Chapter 7: Conclusion |
| 7 | Final Presentation and Demo | (30/8/2021 – 5/9/2021) Presentation to supervisor and evaluator of overall progress and results | Presentation |

### 3.3.3 Project Gantt Chart

Refer to appendix section.

## 3.4 Conclusion

In conclusion, project methodology is a very important phase to ensure the successful development of this project. It also helps to achieve the objectives of this research. All of the phases have been explaining in this chapter, with the approaches and methods chosen. All the approaches and methods are the main goals of this project to measure the accuracy of mobile malware detection. The next chapter will describe more about defined methodology and procedures in this project. The next chapter also will discuss more details in developing machine learning classification using SVM.

# CHAPTER 4: ANALYSIS AND DESIGN

## 4.1 Introduction

In this chapter, the more profound explanation of the design and analysis of the proposed methods. To ensure this project meets its objectives, all the processes involved are identified first and analyzed before going to the next phase. For feature selection, Information Gain and Chi-Square are the best methods to implement in this experiment. Both are chosen from the literature that has a high recommendation. Then, the features go through machine learning algorithms to develop a model. The hardware and software are also evaluated to carry out this research. The design phase will provide a flowchart that illustrates the process involved and shows the relationship. The analysis and design section will give a clear understanding of the project and reduce future mistakes.

## 4.2 Problem Analysis

The main objective of this research is to identify a model of machine learning to detect mobile malware. The detecting process is carried out automatically utilizing a machine learning algorithm that has been pre-programmed. The problem occurs when the dataset is not in a binary format that is both benign and malignant. This research will use an opcode sequence. A big number of instances need high computing resources and high – precision algorithm for machine learning. This project will solve classification of mobile malware. The classifier used in this research is Support Vector Machine (SVM), which is a well-known classification method and the best classification for computer - assisted detection. This is the best technique to solve the problems.

## 4.3 Project Design

The design phase will show there are several phases involved in conducting this research in the future. Therefore, each phase has its function in processing data information to decide according to the objectives of this project. Figure 4.1 shows the illustration of this project design.

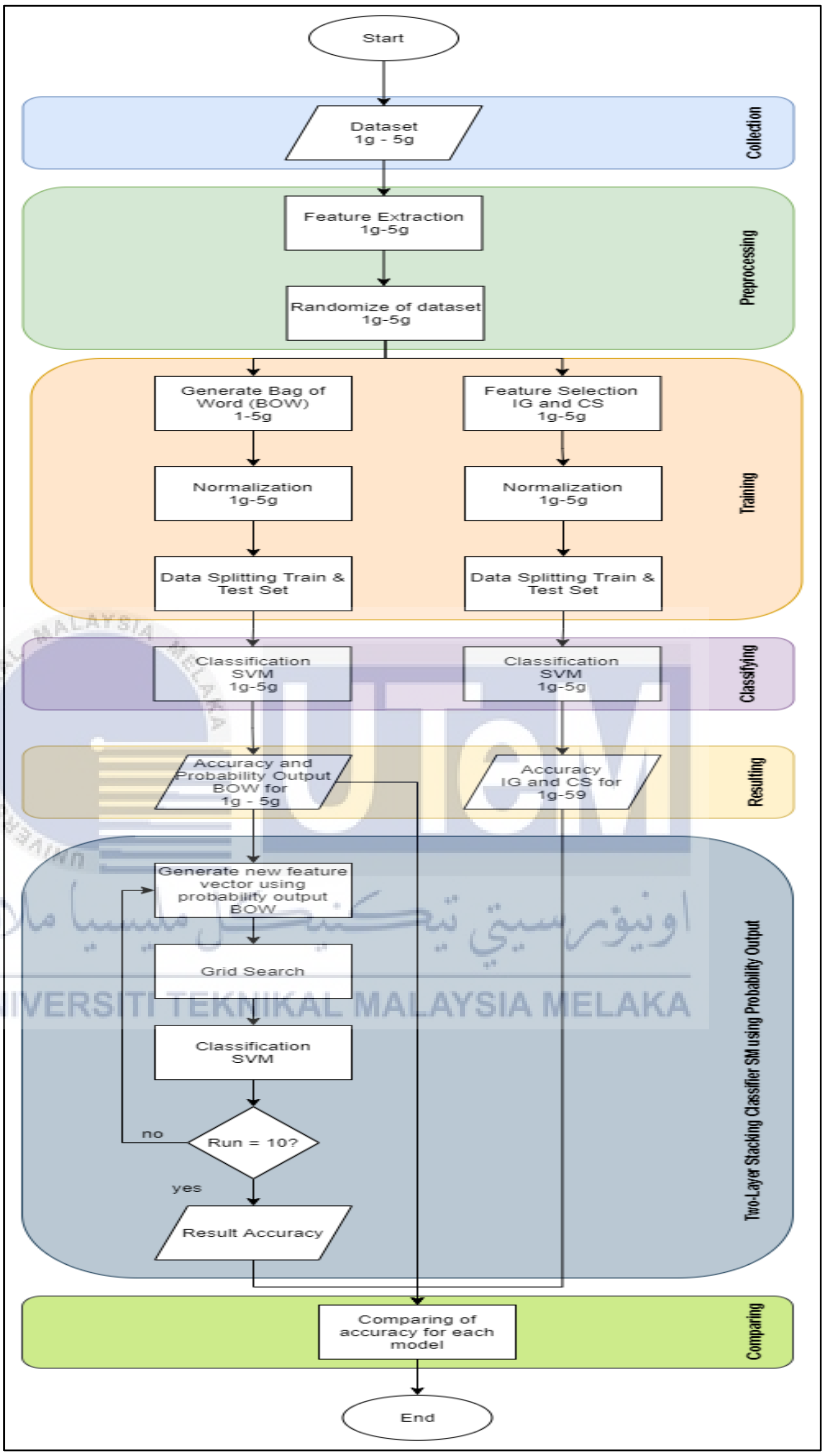

Start
Collection
Dataset 1g - 5g
Preprocessing
Feature Extraction 1g-5g
Randomize of dataset 1g-5g
Training
Generate Bag of Word (BOW) 1-5g
Feature Selection IG and CS 1g-5g
Normalization 1g-5g
Normalization 1g-5g
Data Splitting Train & Test Set
Data Splitting Train & Test Set
Classifying
Classification SVM 1g-5g
Classification SVM 1g-5g
Resulting
Accuracy and Probability Output BOW for 1g - 5g
Accuracy IG and CS for 1g-59
Two-Layer Stacking Classifier SM using Probability Output
Generate new feature vector using probability output BOW
Grid Search
Classification SVM
Run = 10?
no
yes
Result Accuracy
Comparing
Comparing of accuracy for each model
End

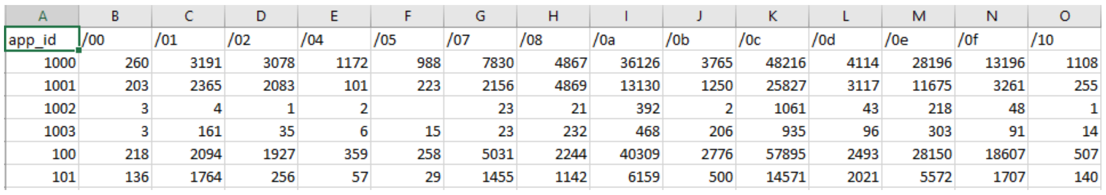
**Figure 4.1: Flowchart Design**

### 4.3.1 Dataset

Dataset is raw data that can be obtained from the previous researcher related to the topic of research. Dataset is required to apply the theory and concept of machine learning algorithms. It also provides information to the machine learning algorithm to detect mobile malware. This research used a dataset from the result of previous research, which is (Anuar, Mas'ud, Bahaman, & Mat Ariff, 2020) and the title of the research is Mobile Malware Behaviour through Opcode Analysis. The dataset is analysed from all datasets from previous research using frequency appearance for opcode in benign applications and malware applications. The dataset is in n-opcode features sequence. Table 4.1 describe the dataset and Figure 4.3 shows the one part from Data_1g dataset. The opcode will refer to Dalvik Opcodes to understand the meaning in each feature.

**Table 4.1: Description of Dataset**

| Dataset | Total Attribute | Total Instance |
|---|---|---|
| 1-gram | 110 | 295 |
| 2-gram | 3634 | 295 |
| 3-gram | 49333 | 295 |
| 4-gram | 253303 | 295 |
| 5-gram | 758126 | 295 |



| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| app_id | /00 | /01 | /02 | /04 | /05 | /07 | /08 | /0a | /0b | /0c | /0d | /0e | /0f | /10 |
| 1000 | 260 | 3191 | 3078 | 1172 | 988 | 7830 | 4867 | 36126 | 3765 | 48216 | 4114 | 28196 | 13196 | 1108 |
| 1001 | 203 | 2365 | 2083 | 101 | 223 | 2156 | 4869 | 13130 | 1250 | 25827 | 3117 | 11675 | 3261 | 255 |
| 1002 | 3 | 4 | 1 | 2 |  | 23 | 21 | 392 | 2 | 1061 | 43 | 218 | 48 | 1 |
| 1003 | 3 | 161 | 35 | 6 | 15 | 23 | 232 | 468 | 206 | 935 | 96 | 303 | 91 | 14 |
| 100 | 218 | 2094 | 1927 | 359 | 258 | 5031 | 2244 | 40309 | 2776 | 57895 | 2493 | 28150 | 18607 | 507 |
| 101 | 136 | 1764 | 256 | 57 | 29 | 1455 | 1142 | 6159 | 500 | 14571 | 2021 | 5572 | 1707 | 140 |

**Figure 4.2: Part of Dataset 1g**

## 4.3.2 Feature Extraction

Feature extraction begins with collecting measured data and creating derived value features that are both informative and non-redundant. Then, from the dataset obtained, the data already extracted by the researcher.

## 4.3.3 Feature Selection

The features gained from the feature extraction are noisy and redundant. So, apply feature selection will remove the irrelevant data and redundancy data. Feature selection also can reduce dimensionality, and calculation cost also reduce overfitting to machine learning. This research will use Information Gain (IG) and Chi-Square (CS) for feature selection to gain performance that affects prediction accuracy.

Weka Application carries out information Gain (IG) feature selection. Weka is a collection of machine learning for data mining that is known as open-source software. It has tools for clustering, classification, regression, and data preparation. The attribute evaluator is used to calculate the information gain (entropy) for each output variable attribute using the Ranker Search method. The value of IG ranges from 0 (no information) to 1 (complete information). The attribute closer to 1 has a higher rank, while the attribute that closes to 0 has a lower rank. Below shows the formula for Entropy and Information Gain (IG).

Entropy:

$$H = -\sum_{i=1}^{k} P_k \log_2 P_k$$

Information Gain:

$$IG = H - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

Where:

k = Class H = Number of inputs $P_k$ = Dataset m = total number of instances

Next, this research also used Chi-Square for the feature selection. It will test and examine whether the target variable depends on the feature variable of numeric data. If they are dependant, they will remain, and if they are independent, the feature variable is considered unimportant and discarded. The target variable, class label, is examined using the chi-square statistic between all feature variables to see a relationship between the target and feature variable. If the class label has an independent relationship to the feature variable, it is discarded. While, if they are dependent, the feature variable is deemed essential because it is directly tied to the result/outcome. To determine if the feature variable and the class label are dependent, a particular cut-off is employed. The null hypothesis and alternate hypothesis in the Chi-square test are as follows:

*H0: Two variables are independent*
*H1: Two variables are dependent*

The formula of Chi-Square feature selection is as follow below:

$$X^2(t,c) = \left[\frac{N \times (WZ - YZ)^2}{(W+X)(W+Z)(W+X)(Y+Z)}\right]$$

Where:

| | |
|---|---|
| W = No. of times feature t and class label c | X = No. of times t |
| Y = No. of times c | No. of times c or t |
| N = Total number of records | |

### 4.3.4 Normalization

Normalization is one of the techniques that responsible for preparing data for machine learning to understand easily. In addition, it will make changes to the numeric column to the range for general use without affecting the values. This project will normalize the frequency in the column to the value range 0 to 1. This research is used min-max normalization method. The formula defined as shown below:

$$y = \frac{(X - \min)}{( - \min)}$$

Where:

min = minimum value of X max = maximum value of X

X = set of observed value

### 4.3.5 Classification

Support Vector Machine (SVM) is a technique for supervised learning that classifies linear and nonlinear data using a nonlinear mapping to translate the original training data into a higher dimension and maximize the margin between support points. In this research, the classification used is SVM. SVM is a supervised machine learning for classification or regression. SVM plot each data as a point using n-dimensional space, which is n referring to number of features with the value of each feature is the value of a coordinate as in Figure 4.3.

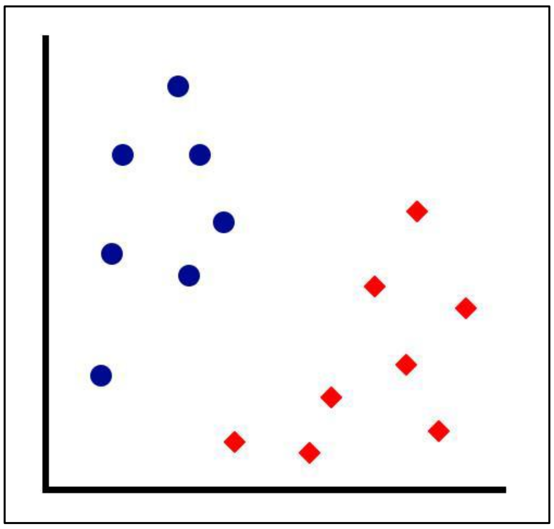


**Figure 4.3: Scatter Plot**

Next, SVM draws a decision line (hyperplane) that separates them and differentiates between two classes, as in Figure 4.4. To draw the hyperplane, there are a few criteria to determine its best position. First, the hyperplane must be drawn as closest as possible for class separation. The hyperplane's closest point is the support vector, which the origin of the SVM name. The hyperplane with the greatest distance from the support vectors is the most optimal, and the distance between the hyperplane and the support vectors is called margin. Figure 4.8 shows the hyperplane with support vectors and the margin.

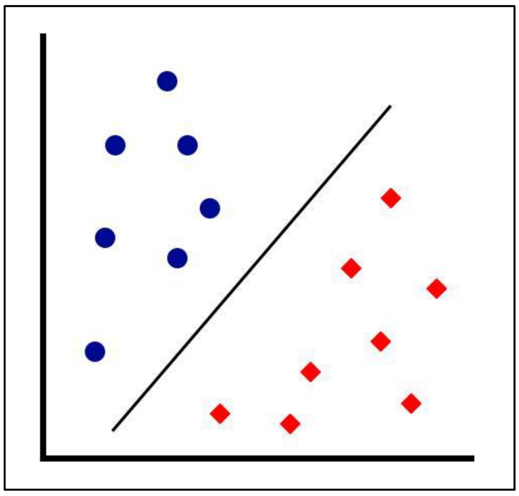


**Figure 4.4: Placement of Hyperplane**

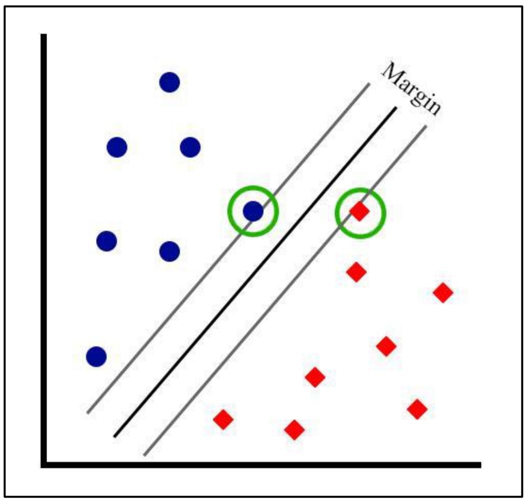


**Figure 4.5: Hyperplane with Margin**

Choosing the correct position for the hyperplane can be difficult. Figure 4.6 below shows the hyperplane 2 classification is an offence classification, while the hyperplane 1 classification is correct. Then, there also has some error of prediction. The accuracy of hyperplane 1 is higher than hyperplane 2 despite having lower margin.

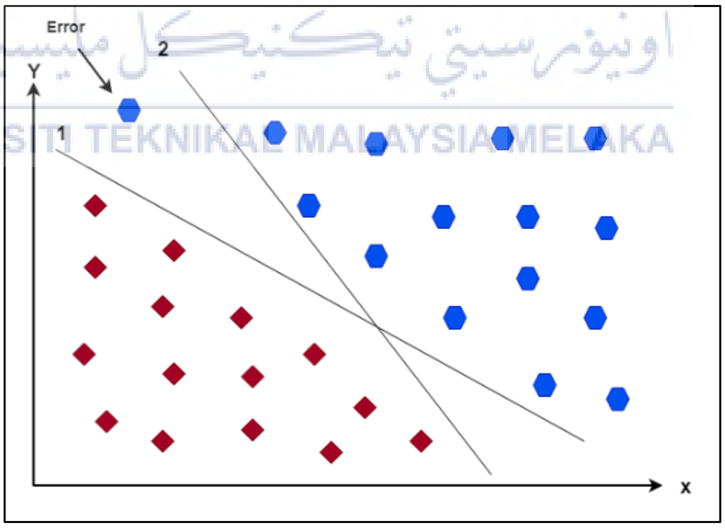


**Figure 4.6: Best Hyperplane**

There are four different combination prediction forms and values in the matrix of confusion. The combinations are true negative (TN), true positive (TP), false negative (FN), and false-positive (FP). True positive is predicted to be positive and identified as true, while false positive is predicted to be positive but identified as false. True negative is predicted to be negative and exactly false, while false negative is predicted to be negative and identified as true. To generate the confusion matrix, the following formula are used:

$$\boldsymbol{Accuracy = \frac{TP+TN}{Total\ Data}} \qquad \boldsymbol{Precision = \frac{TP}{TP+FP}}$$

$$\boldsymbol{Recall = \frac{TP}{TP+FN}} \qquad \boldsymbol{F1\ Score = 2 \times \frac{Precision \times Recall}{Precision \times Recall}}$$

Sometimes, the case in Figure 4.7 below might happen where one of the data lies in other classes, known as an outlier. SVM needs to ignore the outlier and find the best hyperplane with a maximum margin to solve this problem. This is one of the benefits of using SVM where it can solve the problem. The above scenario shows a description of a Linear SVM in which the vector is divided by straight lines.

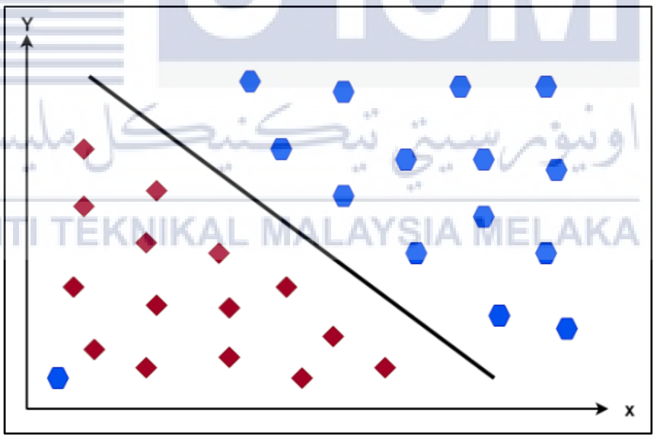


**Figure 4.7: Hyperplane with Outlier**

In real life, the most common problem is the non-linear types as in Figure 4.8. The kernel function in SVM will be used to solve this problem. Kernel will transform non-linear spaces into linear spaces. In a nutshell, it converts the two variables x and y into a new feature space called z. It transforms two-dimensional space into multi-dimensional space as in Figure 4.9. Then, to separate this non-linear data, a hyperplane can be drawn.

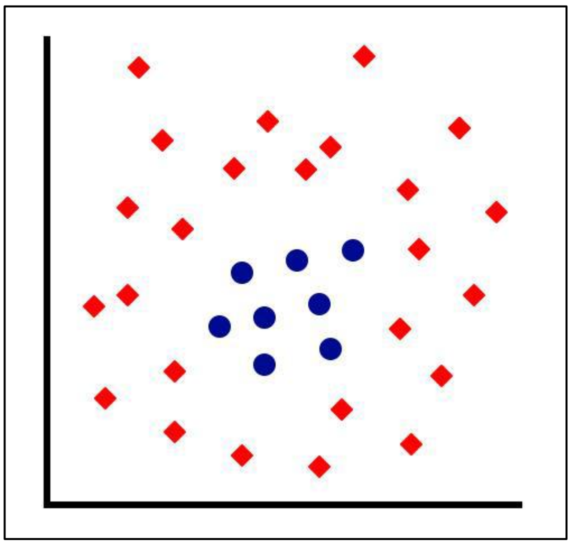


**Figure 4.8: Non-Linear SVM**

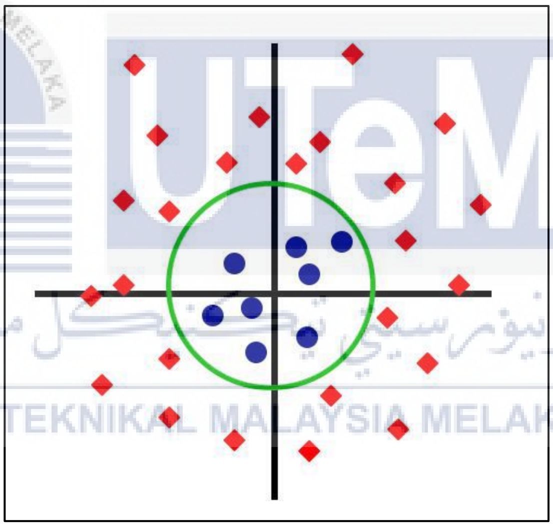


**Figure 4.9: SVM with Multi-Dimensional Space**

A kernel act as crucial role in determining construction of the hyperplane. Table 4.2 shows the comparison on type of SVM's kernel with its functionality.

**Table 4.2: Type of SVM's Kernal Summary**

| Type of Kernel | Description |
|---|---|
| Linear | Vectors separate linearly by using straight line. It good in classifying into two classes at the same time. |
| Polynomial | Performs well with data that non-linearly separable. |
| Radial Basis Function (RBF) | This kernel is used to classify non-linear data. It also can be used for function approximation, series prediction, system control. |
| Sigmoid | This kernel is very popular with its neural network origin. The approach is less time consumption. |

Therefore, all of the kernels are required to test and compare the accuracy of the hyperplane. The high accuracy is chosen and used throughout this project research. This research focuses more on the high accuracy of new model development for detection. Hence, the result of the new model has compared its accuracy with another accuracy by using Chi-square and Information Gain. This is to ensure the new model has effective in the detection of mobile malware.

### 4.3.6 Two-Layer Stacking SVM Classifier using Output Probabilities

This project will implement a deep learning approach known as the two-layer stacking method, or the specific is two-layer stacking SVM classifier using output probabilities. It combines the outputs from the different classifiers of SVM. This approach will reduce a feature vector size and also help improve the performance of the SVM classifier.

Results from the first layer of SVM, which is SVM with BOW, Information Gain, and Chi-Square, will combine. It will convert the outputs to the posterior class probability values. Then, it will learn another SVM classifier, which is for the two-layer.

## 4.4 Requirement Analysis

This section will describe the information of the components involved in the development of the system. Therefore, each criterion will follow this part, which is crucial to make sure the system journey follows the plan proposed in this project research.

### 4.4.1 Software Requirement

There is some software will use in this project to complete the development of the system. It will list with the description of the function of the software. Table below shows the software requirements list with its description.

**Table 4.3: Software Requirement**

| Software | Description |
|---|---|
| Microsoft Windows 10 | Operating system to run project execution |
| Microsoft Word 2016 | Software for project report and documentation |
| Microsoft Excel 2016 | Software for sorting data according to its attribute and instances |
| Microsoft Project | Software for design a Gantt Chart |
| Draw.io | Software for draw a diagram |
| Weka Application | Software for train data |

| | |
|---|---|
| Eclipse IDE Application | Software for implementation of machine learning algorithm in Java |

### 4.4.2 Hardware Requirement

An Ace Aspire ES 14 are used in this research that act as a workstation to run several tasks starting from the experiments to documentations. Table below show the summary of the specification of laptop.

**Table 4.4: Hardware Requirement**

| Specification | Description |
|---|---|
| Processor Type | AMD Quad - Core A4-7210 APU |
| Processor Speed | 1.80 GHz |
| Display Resolution | 1366 x 768 |
| Display Type | HD Display |
| Display Size | 14" |
| Operating System | Windows 10 Pro |
| Operating System Architecture | 64-bit |
| RAM | 4.00 GB DDR3 L |
| Storage | SSD |

This project also uses another workstation with advanced specification to execute the experiment, which is desktop. The specifications shown in Table 4.5 below.

**Table 4.5: Workstation Specification**

| **Specification** | **Description** |
|---|---|
| Processor Type | Intel® Xeon® CPU E5530 |
| Processor Speed | 2.40 GHz |
| Display Resolution | 1920 x 1080 |
| Display Type | HD Display |
| Operating System | Windows 10 Pro |
| Operating System Architecture | 64-bit |
| RAM | 15.0 GB DDR3 |
| Storage | SSD 480GB |

## 4.5 Conclusion

In conclusion, this section explains in detail the research process involved during this project conducted. The analysis part helps to discover a problem and elaborate a project methodology to its component that can solve it. This part is important to make sure the project achieves its objectives follow the process appropriately planned. The design part is transforming the analysis to the procedures on how to complete this research. All of this is done by studying the literature regarding the research domain that acts as a guide to finish this project. The next chapter will describe an implementation of this project according to the analysis and design are done in this chapter.

# CHAPTER 5: EXPERIMENT

## 5.1 Introduction

This chapter focuses on executing the project based on output in the previous chapter, which is analysis and design. An experiment is conducted to achieve the goals of the project, thus solving the problem statements. A comprehensive list of environment setups is required, which is also will cover in full in this chapter.

## 5.2 Software Development Environment Setup

In this implementation phase, the Windows operating system was used as a platform. The Windows OS is installed in laptops and desktops, which can execute some software, scripts, and documentation throughout this project. Next, Eclipse IDE 2021-03 was selected to code scripts. An Eclipse is well-known software with a standard workspace and plug-in framework for a custom environment. It can also support any programming language such as Java, C, C++, PHP, and JavaScript using the plug-in.

The programming language used in this project is Java, which is popular among programmers with its Java Development Kit (JDK). JDK is one of three main technology packages, and the other two are the Java Virtual Machine (JVM), and the Java Runtime Environment (JRE) integrated with Eclipse. The JDK allows programmers to construct Java applications that the JVM and JRE can compile and run.

## 5.3 Process Module

There has certain module that need to completed in order to carried out the experiment. This phase has two modules, which are Data Collection, and Train Data and Test Data. Figure 5.1 shows the modules involved.

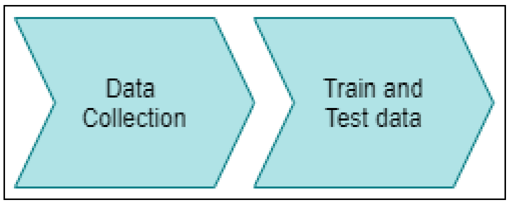


**Figure 5.1: Process Module**

### 5.3.1 Data Collection

The dataset was obtained from previous researches by Zaki Mas'ud who create this mobile malware dataset through opcode sequences. This dataset already labeled their class as malware and benign in N-gram dataset. This project experiment will determine the accuracy of mobile malware attacks using machine learning algorithms. This experiment uses the SVM classifier and Two-Layer Stacking SVM classifier using probabilities output to produce a prediction output.

### 5.3.2 Train and Test Data

In this section, there are five classes: train and test data, model file, prediction file, and test result file. The model is undergoing algorithm code and comes out with output which is the prediction accuracy.

The dataset from 1-gram until 5-gram has the same amount of malware and benign data. In this dataset, all n-gram has 295 instances where there have 95 instances of malware and 200 instances of benign. The dataset will be split into train set and test set. To select instances for test and train, each instance will be randomized its selection to produce train and test data. Figure 5.2 shows the sample of random numbers.

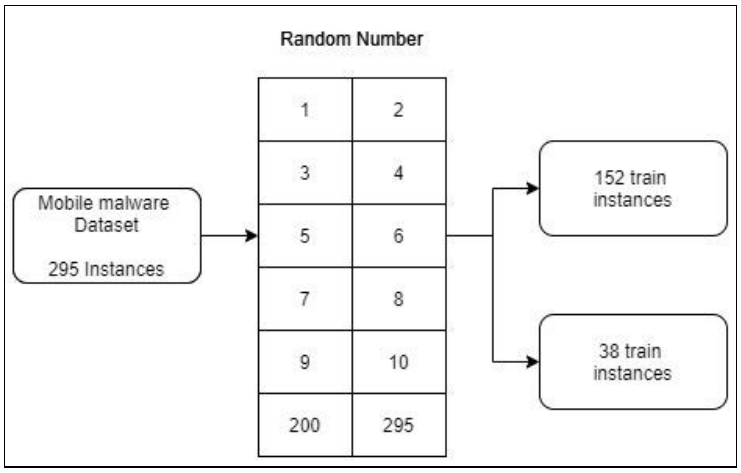
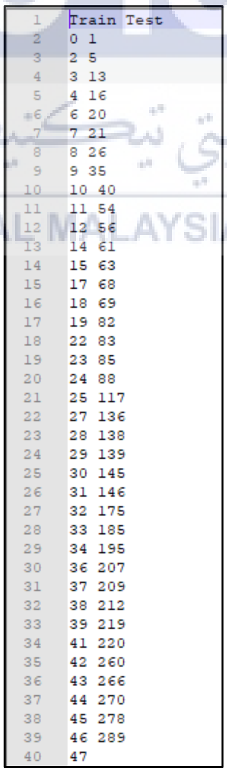


**Figure 5.2: Sample Train and Test Data**

In this experiment, 80% - 20% is used to separate train and test set as in Figure 5.2 and will be used in BOW, IG and CS. Figure 5.3 below shows the random number file that contain two columns represent train data and test data that has been randomized. The train data has 152 instances and 38 instances for test data.



```
Train Test
0 1
2 5
3 13
4 16
6 20
7 21
8 26
9 35
10 40
11 54
12 56
14 61
15 63
17 68
18 69
19 82
22 83
23 85
24 88
25 117
27 136
28 138
29 139
30 145
31 146
32 175
33 185
34 195
36 207
37 209
38 212
39 219
41 220
42 260
43 266
44 270
45 278
46 289
47
```

**Figure 5.3: Example Random File Number**

### 5.3.3 Scale and Model for Train Data

The instances for train and test is used to generate train scale model for feature selection which are CS and IG. The weightage assigned by the train.scale is utilized to generate the prediction file. To create the train, the train.arff file will be used with train.scale and processed in SVM. The following processes are shown in Figure 5.4:

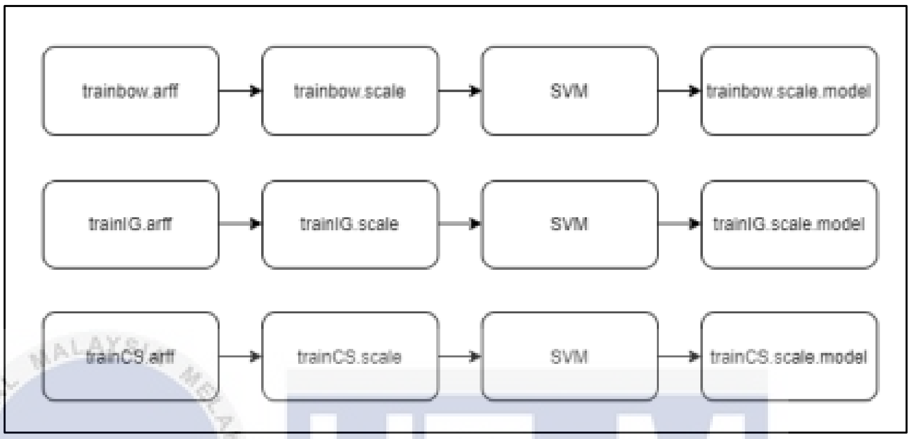


**Figure 5.4: Process creating scale and model for train data**

### 5.3.4 Generate The Predict File

The weighting value of the test.arff file in the prediction is used in train.scale.model. The train.scale.model value will be used by the test.arff file to generate a prediction file. In the prediction file, the occurrences are categorized as 0 for the first class and 1 for the second class. BOW, IG, and CS accuracy are also included in the prediction file. Figure 5.5 shows the process in creating predict file.

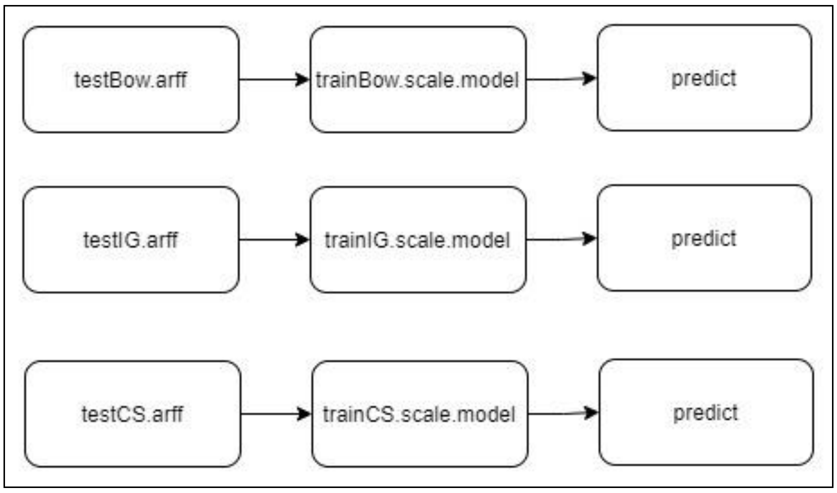
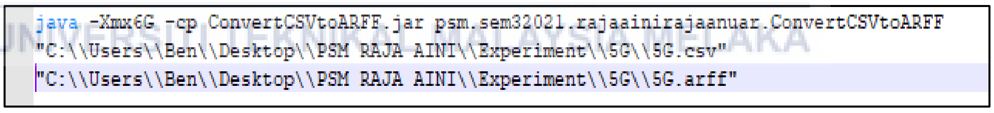


**Figure 5.5: Process generating predict file**

## 5.4 Project Implementation

### 5.4.1 Script Execution

The Java codes are converted to a runnable jar to be run through a batch file program. A batch file consists of parameters that will use to execute the experiment. First, we use this method to convert dataset from CSV to arff file. Figure 5.6 shows the batch file for convert file.



```
java -Xmx6G -cp ConvertCSVtoARFF.jar psm.sem32021.rajaainirajaanuar.ConvertCSVtoARFF
"C:\\Users\\Ben\\Desktop\\PSM RAJA AINI\\Experiment\\5G\\5G.csv"
"C:\\Users\\Ben\\Desktop\\PSM RAJA AINI\\Experiment\\5G\\5G.arff"
```

**Figure 5.6: Convert File Script**

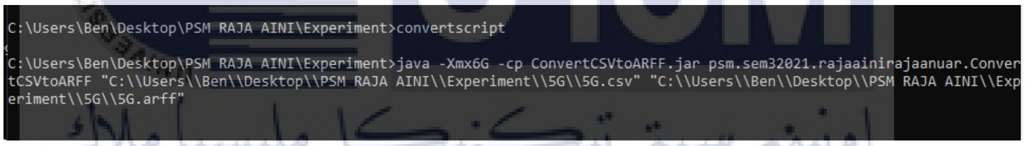
Table 5.1 explain the details on each command used in Figure 5.6.

**Table 5.1: Parameter Description for Batch File**

| Command | Description |
|---|---|
| -Xmx6G | Amount of RAM used |

| | |
|---|---|
| -cp | Execute |
| ConvertCSVtoARFF.jar | The JAR file |
| Psm.sem32021.rajaainirajaanuar.ConvertCSVtoARFF | Packages and class name |
| "C:\\Users\\Ben\\Desktop\\PSM RAJA AINI\\Experiment\\5G\\5G.csv" | The CSV file location |
| "C:\\Users\\Ben\\Desktop\\PSM RAJA AINI\\Experiment\\5G\\5G.arff" | The file location of ARFF file after being converted |

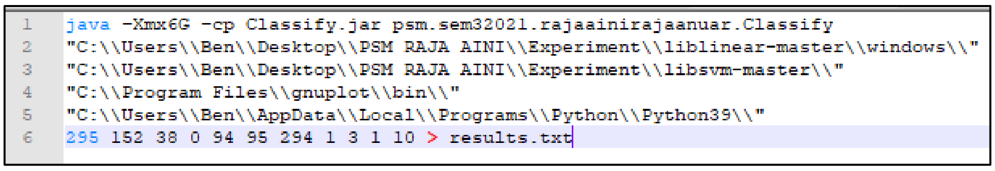
Then, the batch file above is run using command prompt (cmd), were needed to navigate the batch file location and enter the file's name as in Figure 5.7.



```
C:\Users\Ben\Desktop\PSM RAJA AINI\Experiment>convertscript

C:\Users\Ben\Desktop\PSM RAJA AINI\Experiment>java -Xmx6G -cp ConvertCSVtoARFF.jar psm.sem32021.rajaainirajaanuar.ConvertCSVtoARFF "C:\\Users\\Ben\\Desktop\\PSM RAJA AINI\\Experiment\\5G\\5G.csv" "C:\\Users\\Ben\\Desktop\\PSM RAJA AINI\\Experiment\\5G\\5G.arff"
```

**Figure 5.7: Run Batch File Using CMD**

Next, execute code for mobile malware dataset. In this experiment, the execution will be running ten times based on the N-gram. Figure 5.8 below shows the batch file for the mobile malware dataset.



```
java -Xmx6G -cp Classify.jar psm.sem32021.rajaainirajaanuar.Classify
"C:\\Users\\Ben\\Desktop\\PSM RAJA AINI\\Experiment\\liblinear-master\\windows\\"
"C:\\Users\\Ben\\Desktop\\PSM RAJA AINI\\Experiment\\libsvm-master\\"
"C:\\Program Files\\gnuplot\\bin\\"
"C:\\Users\\Ben\\AppData\\Local\\Programs\\Python\\Python39\\"
295 152 38 0 94 95 294 1 3 1 10 > results.txt
```

**Figure 5.8: Sample run batch program for mobile malware dataset**

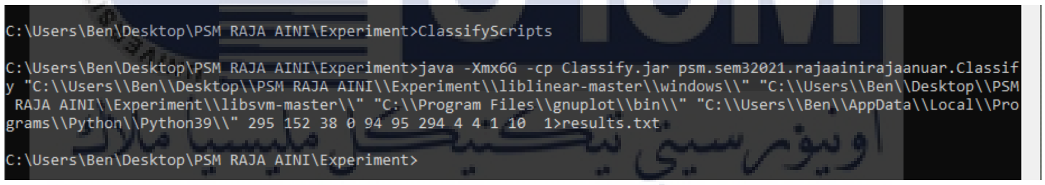
This execution will take some time to complete the whole process based on ten times the run and the size of the gram. The execution also creates several files, as mentioned before, including results, model file, predict file, and scale file based on the parameter and argument set in the batch file. The description of each command is shown in Table 5.2.

**Table 5.2: Description of each command**

| Command | Description |
|---|---|
| -Xmx6G | Amount of RAM used |
| -cp | Execute |
| Classify.jar | The JAR file |
| Psm.sem32021.rajaainirajaanuar.Clasify | Packages and class name |
| \\liblinear-master\\windows | Liblinear path |
| \\libsvm-master | Libsvm path |
| \\gnuplot\\bin | GNU plot path |
| \\Python39 | Python path |
| 295 | Total instances |
| 152 | Total instances for train |
| 38 | Total instances for test |

| | |
|---|---|
| 0 94 | Initial number and range of instances for class 1 |
| 95 294 | Initial number and range of instances for class 0 |
| 1 3 | Dataset in n-gram |
| 1 10 | Start and end run (number of run) |
| results.txt | File created to display the result |

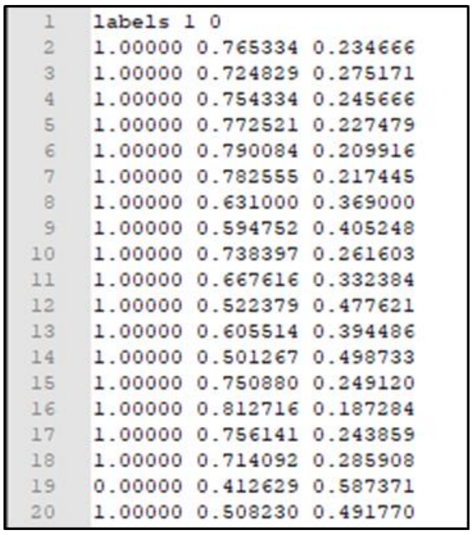
Then, execute above batch file, navigate to the batch file using cmd and enter its file name as in Figure 5.9.



```
C:\Users\Ben\Desktop\PSM RAJA AINI\Experiment>ClassifyScripts

C:\Users\Ben\Desktop\PSM RAJA AINI\Experiment>java -Xmx6G -cp Classify.jar psm.sem32021.rajaainirajaanuar.Classif
y "C:\\Users\\Ben\\Desktop\\PSM RAJA AINI\\Experiment\\liblinear-master\\windows\\" "C:\\Users\\Ben\\Desktop\\PSM
 RAJA AINI\\Experiment\\libsvm-master\\" "C:\\Program Files\\gnuplot\\bin\\" "C:\\Users\\Ben\\AppData\\Local\\Pro
grams\\Python\\Python39\\" 295 152 38 0 94 95 294 4 4 1 10  1>results.txt

C:\Users\Ben\Desktop\PSM RAJA AINI\Experiment>
```

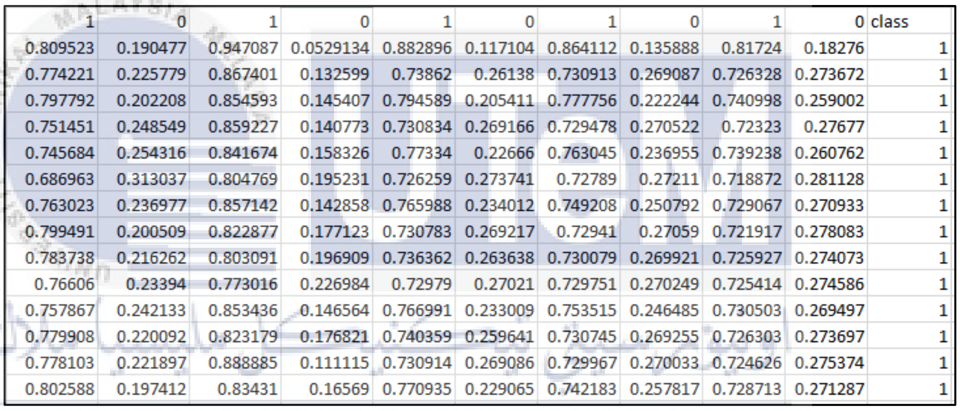
**Figure 5.9: Execute the batch file program**

As previously stated, the above code will run the program for ten iterations and output various files. Next, the execution of second experiment which is Two-layer stacking SVM classifier using probability output. The predict file for all N-gram(1-5gram) from previous experiment are used in this method. BOW result and linear kernel are used for second experiment. Figure 5.10 shows the example predict file which contain probability values for malware and benign. The combination all predict file that become probability output shown in Figure 5.11. The probability output file format is saved as CSV and converted to ARFF format file.



```
labels 1 0
1.00000 0.765334 0.234666
1.00000 0.724829 0.275171
1.00000 0.754334 0.245666
1.00000 0.772521 0.227479
1.00000 0.790084 0.209916
1.00000 0.782555 0.217445
1.00000 0.631000 0.369000
1.00000 0.594752 0.405248
1.00000 0.738397 0.261603
1.00000 0.667616 0.332384
1.00000 0.522379 0.477621
1.00000 0.605514 0.394486
1.00000 0.501267 0.498733
1.00000 0.750880 0.249120
1.00000 0.812716 0.187284
1.00000 0.756141 0.243859
1.00000 0.714092 0.285908
0.00000 0.412629 0.587371
1.00000 0.508230 0.491770
```

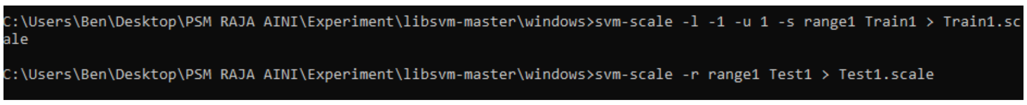
**Figure 5.10: Sample Predict File**



| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | class |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.809523 | 0.190477 | 0.947087 | 0.0529134 | 0.882896 | 0.117104 | 0.864112 | 0.135888 | 0.81724 | 0.18276 | 1 |
| 0.774221 | 0.225779 | 0.867401 | 0.132599 | 0.73862 | 0.26138 | 0.730913 | 0.269087 | 0.726328 | 0.273672 | 1 |
| 0.797792 | 0.202208 | 0.854593 | 0.145407 | 0.794589 | 0.205411 | 0.777756 | 0.222244 | 0.740998 | 0.259002 | 1 |
| 0.751451 | 0.248549 | 0.859227 | 0.140773 | 0.730834 | 0.269166 | 0.729478 | 0.270522 | 0.72323 | 0.27677 | 1 |
| 0.745684 | 0.254316 | 0.841674 | 0.158326 | 0.77334 | 0.22666 | 0.763045 | 0.236955 | 0.739238 | 0.260762 | 1 |
| 0.686963 | 0.313037 | 0.804769 | 0.195231 | 0.726259 | 0.273741 | 0.72789 | 0.27211 | 0.718872 | 0.281128 | 1 |
| 0.763023 | 0.236977 | 0.857142 | 0.142858 | 0.765988 | 0.234012 | 0.749208 | 0.250792 | 0.729067 | 0.270933 | 1 |
| 0.799491 | 0.200509 | 0.822877 | 0.177123 | 0.730783 | 0.269217 | 0.72941 | 0.27059 | 0.721917 | 0.278083 | 1 |
| 0.783738 | 0.216262 | 0.803091 | 0.196909 | 0.736362 | 0.263638 | 0.730079 | 0.269921 | 0.725927 | 0.274073 | 1 |
| 0.76606 | 0.23394 | 0.773016 | 0.226984 | 0.72979 | 0.27021 | 0.729751 | 0.270249 | 0.725414 | 0.274586 | 1 |
| 0.757867 | 0.242133 | 0.853436 | 0.146564 | 0.766991 | 0.233009 | 0.753515 | 0.246485 | 0.730503 | 0.269497 | 1 |
| 0.779908 | 0.220092 | 0.823179 | 0.176821 | 0.740359 | 0.259641 | 0.730745 | 0.269255 | 0.726303 | 0.273697 | 1 |
| 0.778103 | 0.221897 | 0.888885 | 0.111115 | 0.730914 | 0.269086 | 0.729967 | 0.270033 | 0.724626 | 0.275374 | 1 |
| 0.802588 | 0.197412 | 0.83431 | 0.16569 | 0.770935 | 0.229065 | 0.742183 | 0.257817 | 0.728713 | 0.271287 | 1 |

**Figure 5.11: Probability Output**

Scale files need to be generated before the train process start. Each feature of the training data should be scaled to [-1,1]. The file range is used to contain scaling factors, which are then required to scale the test data. This execution is run 10 times. The command shows in the Figure 5.12 below and the description for each command as in Table 5.3.



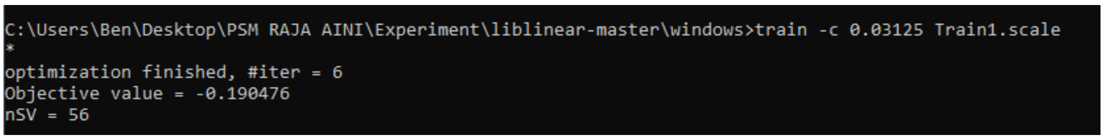
```
C:\Users\Ben\Desktop\PSM RAJA AINI\Experiment\libsvm-master\windows>svm-scale -l -1 -u 1 -s range1 Train1 > Train1.sc
ale

C:\Users\Ben\Desktop\PSM RAJA AINI\Experiment\libsvm-master\windows>svm-scale -r range1 Test1 > Test1.scale
```

**Figure 5.12: Sample execution for scaling**

**Table 5.3: The Command and Its Description**

| Command | Description |
|---|---|
| svm-scale | Tool for scaling input data file |
| -l -1 | Scaling lower limit default -1 |
| -u 1 | Scaling upper limit default 1 |
| -s range1 | Save scaling parameters to range1 file name |
| Train1.scale | File created to display the scaling for train |
| -r range1 | Restore scaling parameter from range1 file |
| Test1.scale | File created to display the scaling for test |

Next, by using L2-loss SVM for cross validation, determine the value C that achieves the best cross validation accuracy. Then, using the chosen C, train the data to create a model. The command and the details are explained as in Figure 5.13 and Table 5.4.



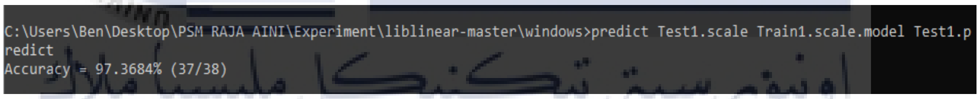
```
C:\Users\Ben\Desktop\PSM RAJA AINI\Experiment\liblinear-master\windows>train -c 0.03125 Train1.scale
*
optimization finished, #iter = 6
Objective value = -0.190476
nSV = 56
```

**Figure 5.13: The Command for Train Data**

**Table 5.4: The Description of Comment for Train Data**

| Command | Description |
|---|---|
| train | Tool for train data |
| -c 0.03125 | Set the parameter of C of C-SVC, epsilon-SVR, and nu-SVR |
| Train1.scale | Data file name |

After train the probability output, this experiment run a testing on test data based on training to get the accuracy. This execution is run through command prompt. The command for testing shown in Figure 5.14. Table 5.5 shows the explanation of each command.



```
C:\Users\Ben\Desktop\PSM RAJA AINI\Experiment\liblinear-master\windows>predict Test1.scale Train1.scale.model Test1.p
redict
Accuracy = 97.3684% (37/38)
```

**Figure 5.14: Testing Command**

**Table 5.5: Testing's Command Description**

| Command | Description |
|---|---|
| Predict | Tool to test data |
| Test1.scale | Testing's scale file |
| Train1.scale.model | Train model file |

| | |
|---|---|
| Test1.predict | File created to display the predict result |

## 5.5 Result

In this project’s experiment, there has execute two phase experiments, which is first experiment is using SVM classifier and for second experiment is using Two-Layer Stacking SVM classifier using predict file as a probability output. Each experiment come out with the best prediction accuracy. This experiment has three types of model results, which are BOW, feature selection Information Gain, and Chi-Square. The dataset is in N-gram range from 1-gram to 5-gram for this experiment. The kernels used in this project are Linear and RBF. The total instances of the dataset are 295, where 95 is malware and 200 are benign.

First experiment executes BOW, feature selection Information Gain and Chi-Square in Support Vector Machine classifier.

### 5.5.1 Attribute

The total instances for this project is 295 to generate BOW accuracy for all N-gram. While for attribute, it was selected by using ranking method in feature selection by referring to the level; of relevance for prediction. In previous chapter stated for attribute selection, this project uses 80% of total number of attributes. Table 5.3 below shows the total number used in this project.

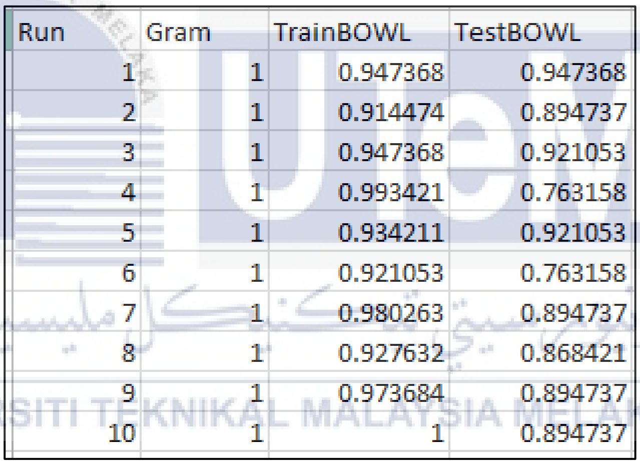
**Table 5.6: The Total Number of Attributes**

| Gram | Total Attributes | 80% of Total Attribute |
|---|---|---|
| 1 | 110 | 88 |
| 2 | 3634 | 2907 |

| | | |
|---|---|---|
| 3 | 49333 | 39466 |
| 4 | 253303 | 20642 |
| 5 | 758126 | 606500 |

### 5.5.2 10 runs

BOW, IG, and CS for each kernel execute 10 run times. This run time is applied to 1-gram until 5-gram same as with second experiment. Figure shows the sample of this experiment run in 10 cycles.



| Run | Gram | TrainBOWL | TestBOWL |
|---|---|---|---|
| 1 | 1 | 0.947368 | 0.947368 |
| 2 | 1 | 0.914474 | 0.894737 |
| 3 | 1 | 0.947368 | 0.921053 |
| 4 | 1 | 0.993421 | 0.763158 |
| 5 | 1 | 0.934211 | 0.921053 |
| 6 | 1 | 0.921053 | 0.763158 |
| 7 | 1 | 0.980263 | 0.894737 |
| 8 | 1 | 0.927632 | 0.868421 |
| 9 | 1 | 0.973684 | 0.894737 |
| 10 | 1 | 1 | 0.894737 |

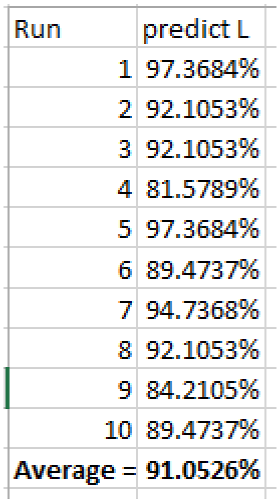
**Figure 5.15: 10 Run Time**

### 5.5.3 Accuracy Table

Table below shows the accuracy for single classifier SVM with Bag of Word, feature selection Information Gain and, Chi-Square. This is average result in testing for 10 run time for both kernels. Based on the accuracy table below, the high percentage is 91.84 in 4-gram Linear Kernel for BOW, IG and CS. 2-gram also has high accuracy in Linear kernel for IG.

| Single Classifier | | | | | | |
|---|---|---|---|---|---|---|
| Gram | Bag of Word | | Information Gain | | Chi-Square | |
| | Linear | RBF | Linear | RBF | Linear | RBF |
| 1 | 87.63 | 87.58 | 85.79 | 87.37 | 87.63 | 86.84 |
| 2 | 91.32 | 87.37 | 91.84 | 88.95 | 91.32 | 87.89 |
| 3 | 91.05 | 88.68 | 91.32 | 90.00 | 91.05 | 87.89 |
| 4 | 91.84 | 87.12 | 91.84 | 90.26 | 91.84 | 87.11 |
| 5 | 91.58 | 52.37 | 91.58 | 70.26 | 91.58 | 52.11 |

For second experiment, the BOW probability output is used because it has best accuracy from previous experiment. The chosen kernel is Linear. The predict files from first experiment from 1-5gram are combined and become probability output. All attributes are used in this execution. Figure 5.16 below shows the accuracy table for second experiment approach which is two-layer stacking SVM classifier using probability output. The probability outputs are run 10 time and the average of the accuracy are calculated to get the final result.



| Run | predict L |
|---|---|
| 1 | 97.3684% |
| 2 | 92.1053% |
| 3 | 92.1053% |
| 4 | 81.5789% |
| 5 | 97.3684% |
| 6 | 89.4737% |
| 7 | 94.7368% |
| 8 | 92.1053% |
| 9 | 84.2105% |
| 10 | 89.4737% |
| **Average =** | **91.0526%** |

**Figure 5.16: Result Second Experiment**

A graph bar for experiment 1 and experiment 2 are created from the table result as in Figure 5.17 and Figure 5.18. For the first experiment graph, the average is calculated and display it by pass it to a custom table.

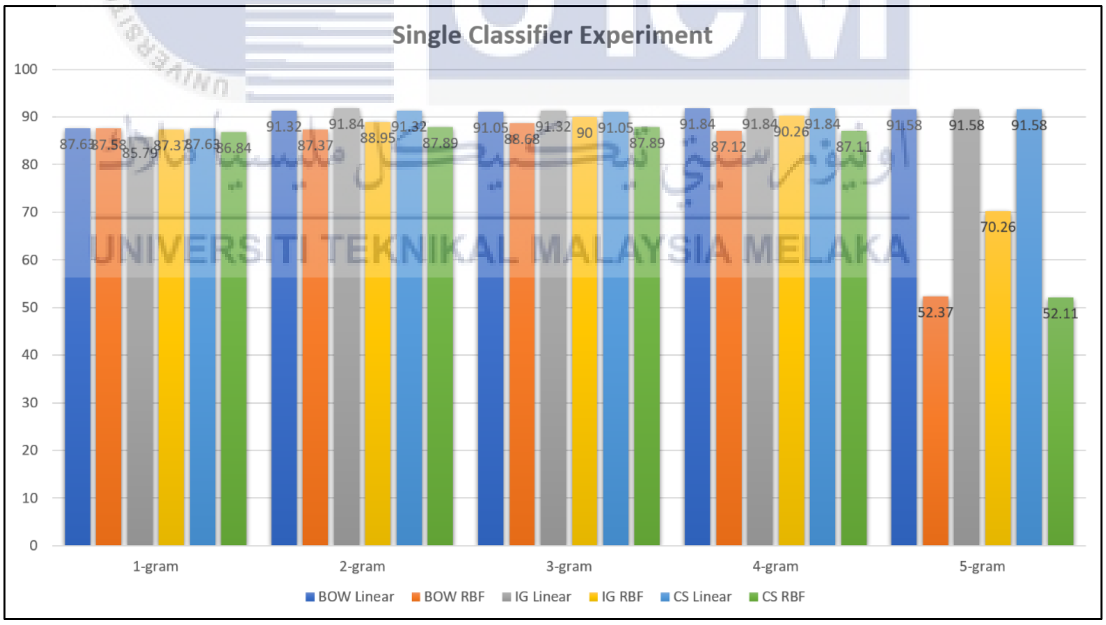


**Figure 5.17: Bar Graph First Experiment**

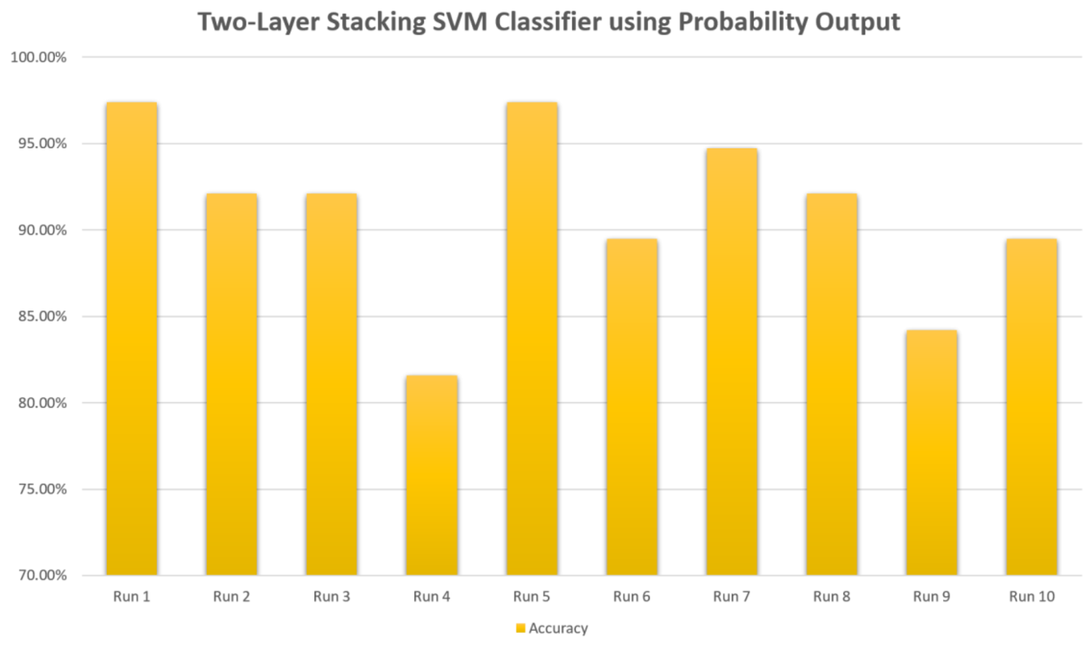


**Figure 5.18: Second Experiment Bar Graph**

# CHAPTER 6: DISCUSSION

## 6.1 Introduction

The previous chapter explained about the details implementation including environment setup and software involved, the process, modules involved, execution and lastly the result of the experiment. The result was recorded in table and displayed in graph to better understanding and summarize the results.

This section will explain about the analysis of this project from the first step until completed this project. The accuracy is evaluated either it can pass the benchmark or not. The comparison between previous experiment and new proposed method also stated in this section.

## 6.2 Discussion of The Project

The growth of technology in mobile devices is exposed the malware attack that user can get with or without their knowledge. This research has two phases for detect a mobile malware or benign. The first phase is data collection, which about the processing the dataset. The dataset is obtained from the previous research. The second phase is train and test dataset which is generating BOW, feature selection IG and feature selection CS.

The first phase is about processing the dataset through label their class either malware or benign. Then, the dataset is converted to ARFF file format so that the machine learning can process it. Next phase, the dataset is undergoing the data splitting to train and, test data. The data is trained and tested using BOW, IG and CS algorithms. After these feature selection algorithms completed, the classifier SVM applied to distinguish between malware and benign and generate prediction. From the predict file, it will evaluate for each operation and each result will be measured.

## 6.3 Discussion on The Newly Proposed Method

Table 6.1 below shows the benchmark on the previous research.

**Table 6.1: Benchmark**

| Feature Selection Method | Accuracy (%) |
|---|---|
| None | 96.2 |
| Chi-Square | 96.0 |
| Information Gain | 96.0 |
| ReliefF | 95.0 |
| Wrapper method + BF | 99.0 |

The newly proposed method did not beat the benchmark as in Table 6.2 which is has a few different. The benchmark feature selections are using SVM linear. There has 4.16% different for Information Gain and Chi-Square method. The factor that influences the result was the data only limit to the content. Then. The huge dataset can cause the overfitting. This final result is for first layer of SVM classifier.

**Table 6.2: Final Result**

| Feature Selection Method | Gram | Accuracy (%) |
|---|---|---|
| Bag Of Word | 4 | 91.84 |
| Information Gain | 2,4 | 91.84 |

| | | |
|---|---|---|
| Chi-Square | 4 | 91.84 |

The two-layer stacking SVM classifier using probability output for this experiment is using BOW linear prediction data. This is because of result from the single classifier shown in previous chapter that BOW linear has a higher average accuracy for all N-gram. This method is for feature vector and weighting scheme to improve the accuracy. However, this approach did not outperform the accuracy from the first layer SVM classifier. Figure 6.1 below shows the final accuracy that this method achieved.

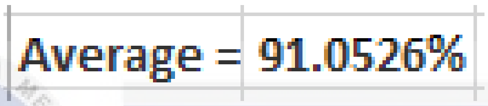

Average = 91.0526%

**Figure 6.1: Final Accuracy**

## 6.4 Conclusion

In conclusion, this chapter is about the management of implementation, execution and analysis of this project. This chapter shows all the procedure in this experiment conducted. As the proposed models did not outperform the benchmark and did not beat the single classifier, the factors were stated in this chapter. This method can get the best result depending on the setting.

# CHAPTER 7: PROJECT CONCLUSION

## 7.1 Introduction

In this section, the summarization of the whole project including all results and findings. This chapter is essential to ensure the project's objectives are achieved successfully and discuss in detail how each objective reaches its success. The constraints and the future improvement are also discussed and analyzed the main factor affecting this direction of the project.

## 7.2 Project Summary

The number of mobile devices users is rapidly increasing because it can be handled with one hand and can control any wireless device, even the mobile far. This technology is attractive to an intruder to attack mobile devices, and this thing needs to avoid before any lost data or damage occur. To prevent any mobile malware attack, the taxonomy and how it occurs need to fully understand and examine the behavior and get the best ways to detect them using a machine learning approach.

This project experiment has produced a tool to differentiate between mobile malware and benign using machine learning algorithms. From the machine learning algorithm, the accuracy model as output and can compare the behavior of attack that undergo bag of words and feature selection information gain and chi-square. Then go through the SVM classifier to get the model accuracy. The result probabilities are combined and enter into the SVM classifier again to get a better result. This method is known as a two-layer stacking SVM classifier. This process distinguishes whether the result can or not outperforms the benchmark.

## 7.3 Project Contribution

The contribution throughout this project is proposed a model for detection malware attack on the mobile devices. This project also proves the proposed method can influence the accuracy.

## 7.4 Project Limitation

This research project only focuses on processing content. The experiment is time-consuming and exhausting because it has a large dataset. This experiment also heavy because this project needs advanced workstation for implementation phase like size of RAM and CPU. Lastly, this research is implementing the machine learning method only.

## 7.5 Future Work

Whoever continued this research can enhance the results and findings of this project. Every research finding, there have several things that need to be improved with future work. As mentioned previously in project constraints, this research can be improved by using another classifier to differentiate which classifier is the best accuracy can be achieved. In data splitting, using various ways like 50-50, 80-20, and k-fold. So, it can be compared which one is the best method.

Another enhancement is executing this project using an advanced specification device to prevent any time-consuming and avoid any damage on the device because of the overload process. Next, another crucial future work was trying another platform, for example, using a Google Colab where it was using python and recommended for deep learning.

## 7.6 Conclusion

In a nutshell, the project research achieved to meet all three objectives successfully. However, there has some highlight future improvement that can be use to develop tools. Any proposed method needs to ensure the high accuracy to be maintain for detection of mobile malware. It will help improve the security of mobile devices.

# REFERENCES

Abdullah, A., Veltkamp, R. C., & Wiering, M. A. (2009). Spatial pyramids and two-layer stacking SVM classifiers for image categorization: A comparative study. *Proceedings of the International Joint Conference on Neural Networks*, 5–12. https://doi.org/10.1109/IJCNN.2009.5178743

Anuar, N. A., Mas'ud, M. Z., Bahaman, N., & Ariff, N. A. M. (2020). Analysis of Machine Learning Classifier in Android Malware Detection through Opcode. *2020 IEEE Conference on Application, Information and Network Security, AINS 2020*, 7–11. https://doi.org/10.1109/AINS50155.2020.9315060

Anuar, N. A., Mas'ud, M. Z., Bahaman, N., & Mat Ariff, N. A. (2020). Mobile Malware Behavior through Opcode Analysis. *International Journal of Communication Networks and Information Security*, *12*(3), 345–354.

Anukrishna, P. R., & Paul, V. (2017). A review on feature selection for high dimensional data. *Proceedings of the International Conference on Inventive Systems and Control, ICISC 2017*, *5*(6), 395–402. https://doi.org/10.1109/ICISC.2017.8068746

Arshad, S., Shah, M. A., Wahid, A., Mehmood, A., Song, H., & Yu, H. (2018). SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System. *IEEE Access*, *6*, 4321–4339. https://doi.org/10.1109/ACCESS.2018.2792941

Dey, A. (2016). Machine Learning Algorithms: A Review. *International Journal of Computer Science and Information Technologies*, *7*(3), 1174–1179. www.ijcsit.com

F.Y, O., J.E.T, A., O, A., J. O, H., O, O., & J, A. (2017). Supervised Machine Learning Algorithms: Classification and Comparison. *International Journal of Computer Trends and Technology*, *48*(3), 128–138. https://doi.org/10.14445/22312803/ijctt-v48p126

Galib, A. H., & Mainul Hossain, B. M. (2019). A Systematic Review on Hybrid Analysis using Machine Learning for Android Malware Detection. *ICIET 2019 - 2nd International Conference on Innovation in Engineering and Technology, December 2019*. https://doi.org/10.1109/ICIET48527.2019.9290548

Gyamfi, N. K., & Owusu, E. (2019). Survey of Mobile Malware Analysis, Detection Techniques and Tool. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON 2018*, *May*, 1101–1107. https://doi.org/10.1109/IEMCON.2018.8614895

Kalpana, S., & Karthikeyan, S. (2018). A survey on rise of mobile malware and detection methods. *Proceedings of 2017 International Conference on Innovations in Information, Embedded and Communication Systems, ICIIECS 2017*, *2018-Janua*(June), 1–5. https://doi.org/10.1109/ICIIECS.2017.8276158

Kouliaridis, V., Barmpatsalou, K., Kambourakis, G., & Chen, S. (2020). A survey on mobile malware detection techniques. *IEICE Transactions on Information and Systems*, *E103D*(2), 204–211. https://doi.org/10.1587/transinf.2019INI0003

Malhotra, A., & Bajaj, K. (2016). A Survey on Various Malware Detection Techniques on Mobile Platform. *International Journal of Computer Applications*, *139*(5), 15–20. https://doi.org/10.5120/ijca2016909159

Mas'ud, M. Z., Sahib, S., Abdollah, M. F., Selamat, S. R., & Huoy, C. Y. (2017). A comparative study on feature selection method for N-gram mobile malware detection. *International Journal of Network Security*, *19*(5), 727–733. https://doi.org/10.6633/IJNS.201709.19(5).10

Sharmeen, S., Huda, S., Abawajy, J. H., Ismail, W. N., & Hassan, M. M. (2018). Malware Threats and Detection for Industrial Mobile-IoT Networks. *IEEE Access*, *6*, 15941–15957. https://doi.org/10.1109/ACCESS.2018.2815660

Singh, L., & Hofmann, M. (2018). Dynamic behavior analysis of android applications for malware detection. *ICCT 2017 - International Conference on Intelligent Communication and Computational Techniques*, *2018-Janua*(March), 1–7. https://doi.org/10.1109/INTELCCT.2017.8324010

Soofi, A., & Awan, A. (2017). Classification Techniques in Machine Learning: Applications and Issues. *Journal of Basic & Applied Sciences*, *13*(September), 459–465. https://doi.org/10.6000/1927-5129.2017.13.76

Tam, K., Feizollah, A., Anuar, N. B., Salleh, R., & Cavallaro, L. (2017). The evolution of android malware and android analysis techniques. *ACM Computing Surveys*, *49*(4). https://doi.org/10.1145/3017427

Venkatesh, B., & Anuradha, J. (2019). A review of Feature Selection and its methods. *Cybernetics and Information Technologies*, *19*(1), 3–26.

https://doi.org/10.2478/CAIT-2019-0001

Wang, Shanshan, Chen, Z., Yan, Q., Yang, B., Peng, L., & Jia, Z. (2019). A mobile malware detection method using behavior features in network traffic. *Journal of Network and Computer Applications*, *133*(December 2018), 15–25. https://doi.org/10.1016/j.jnca.2018.12.014

Wang, Shanshan, Yan, Q., Chen, Z., Yang, B., Zhao, C., & Conti, M. (2017). TextDroid: Semantics-based detection of mobile malware using network flows. *2017 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2017*, 18–23. https://doi.org/10.1109/INFCOMW.2017.8116346

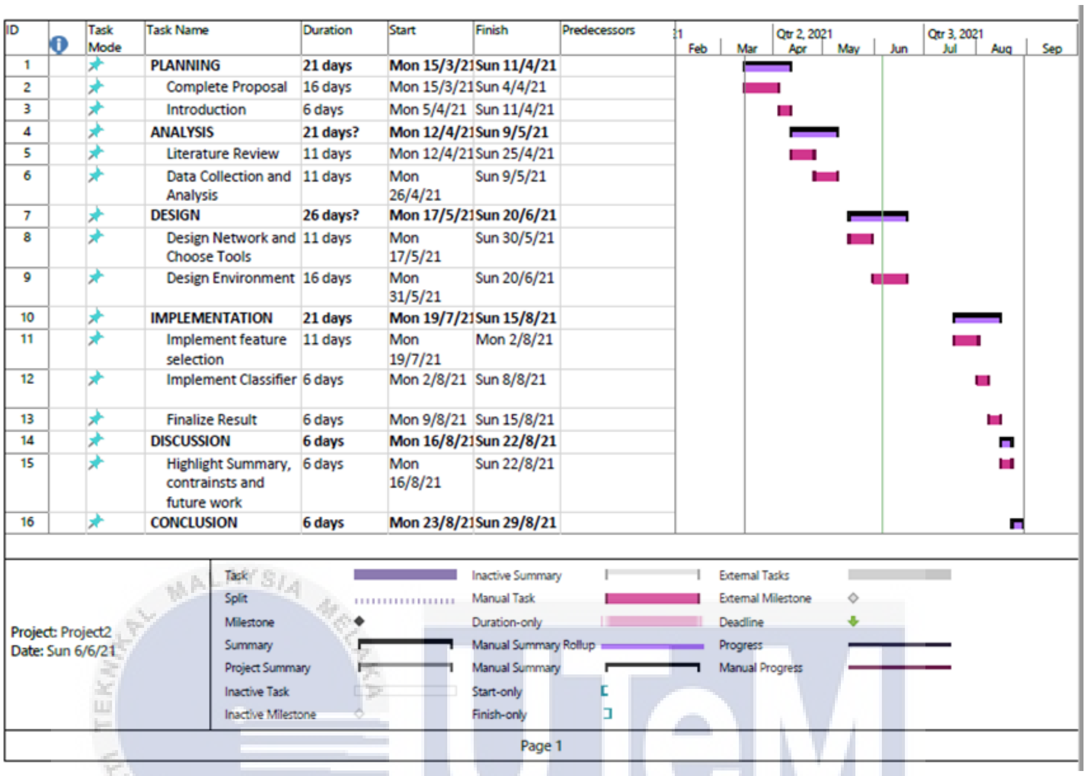
Wang, Suhang, Tang, J., & Liu, H. (2016). Encyclopedia of Machine Learning and Data Mining. *Encyclopedia of Machine Learning and Data Mining*, *January*. https://doi.org/10.1007/978-1-4899-7502-7

Yan, P., & Yan, Z. (2018). A survey on dynamic mobile malware detection. *Software Quality Journal*, *26*(3), 891–919. https://doi.org/10.1007/s11219-017-9368-4

{Bibliography}

# APPENDIX

## A. Gantt Chart



| ID | Task Mode | Task Name | Duration | Start | Finish | Predecessors |
|---|---|---|---|---|---|---|
| 1 | | **PLANNING** | **21 days** | **Mon 15/3/21** | **Sun 11/4/21** | |
| 2 | | Complete Proposal | 16 days | Mon 15/3/21 | Sun 4/4/21 | |
| 3 | | Introduction | 6 days | Mon 5/4/21 | Sun 11/4/21 | |
| 4 | | **ANALYSIS** | **21 days?** | **Mon 12/4/21** | **Sun 9/5/21** | |
| 5 | | Literature Review | 11 days | Mon 12/4/21 | Sun 25/4/21 | |
| 6 | | Data Collection and Analysis | 11 days | Mon 26/4/21 | Sun 9/5/21 | |
| 7 | | **DESIGN** | **26 days?** | **Mon 17/5/21** | **Sun 20/6/21** | |
| 8 | | Design Network and Choose Tools | 11 days | Mon 17/5/21 | Sun 30/5/21 | |
| 9 | | Design Environment | 16 days | Mon 31/5/21 | Sun 20/6/21 | |
| 10 | | **IMPLEMENTATION** | **21 days** | **Mon 19/7/21** | **Sun 15/8/21** | |
| 11 | | Implement feature selection | 11 days | Mon 19/7/21 | Mon 2/8/21 | |
| 12 | | Implement Classifier | 6 days | Mon 2/8/21 | Sun 8/8/21 | |
| 13 | | Finalize Result | 6 days | Mon 9/8/21 | Sun 15/8/21 | |
| 14 | | **DISCUSSION** | **6 days** | **Mon 16/8/21** | **Sun 22/8/21** | |
| 15 | | Highlight Summary, contrainsts and future work | 6 days | Mon 16/8/21 | Sun 22/8/21 | |
| 16 | | **CONCLUSION** | **6 days** | **Mon 23/8/21** | **Sun 29/8/21** | |

Project: Project2
Date: Sun 6/6/21

Page 1