MACHINE LEARNING MALWARE DETECTION FOR ANDROID

AMIR MUHAMMAD HAFIZ BIN OTHMAN

BACHELOR OF COMPUTER SCIENCE
(SOFTWARE ENGINEERING)
WITH HONOURS

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : AMIR MUHAMMAD HAFIZ BIN OTHMAN

Date of Birth

Title : MACHINE LEARNING MALWARE DETECTION FOR ANDROID

Academic Session : SEMESTER II 2022/2023

I declare that this thesis is classified as:

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

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

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

Certified by:

_____ (Student's Signature)

_____ _ New IC/Passport Number Date: 12/6/2023

_____ (Supervisor's Signature)

_____ Name of Supervisor Date: 10/7/23

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

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name	AMIR MUHAMMAD HAFIZ BIN OTHMAN
Thesis Title	MACHINE LEARNING MALWARE DETECTION FOR ANDROID

Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,

_____
(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

SUPERVISOR'S DECLARATION

I/We* hereby declare that I/We* have checked this thesis/project* and in my/our* opinion, this thesis/project* is adequate in terms of scope and quality for the award of the degree of *Doctor of Philosophy/ Master of Engineering/ Master of Science in ……..

_____

(Supervisor's Signature)

Full Name :

Position :

Date :

_____

(Co-supervisor's Signature)

Full Name :

Position :

Date :

STUDENT'S DECLARATION

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

(Student's Signature)

Full Name : AMIR MUHAMMAD HAFIZ BIN OTHMAN

ID Number : CB20152

Date : 12/6/2023

MACHINE LEARNING MALWARE DETECTION FOR ANDROID

AMIR MUHAMMAD HAFIZ BIN OTHMAN

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Doctor of Philosophy/Master of Science/Master of Engineering

Faculty of Computing
UNIVERSITI MALAYSIA PAHANG

OCTOBER 2022

ACKNOWLEDGEMENTS

In accomplishing this research completely, many people have given their best and support day and night. I want to thank those who have been supporting me in finishing this research.

Firstly, I would like to give my greatest thank to Allah for giving me a healthy body and mind to complete this research. Next, I would like to thank my supervisor, Ts. Dr Mohd Faizal Bin Abd Razak continues giving me advice and guidance in completing the research. His guidance is one of the major contributions towards this research.

Then, I would like to thank both of my parents for giving me moral support and especially financial support. They are one of the people who are concerned about me whenever I'm in trouble. I also want to express my thanks to all of my friends for teaching me how to execute the system properly. I want to express my gratitude toward Nazrul Haikal Nazrul Zailani because he is always there for me and is one of my best friends who are concerned about me. With his support, I can give my best for this project.

Lastly, I would like to thank all of the people who helps me indirectly or directly in completing this research successfully.

ABSTRAK

Sepanjang beberapa tahun yang lepas sehingga kini, jumlah perisian perosak yang melakukan kerosakan terhadap sistem operasi Android telah meningkat berbanding system operasi yang lain. Oleh itu, aplikasi yang menggunakan android mesti dianaisis bagi mengesan perisian perosak sementara ia tidak melakukan kerosakan yang serius. Untuk melakukan analisis terhadap aplikasi tersebut, dua jenis analisis perisan perosak boleh digunakan iaitu analisis statik dan analisis dinamik. Analisis statik adalah analisis yang mengkaji kod dalam aplikasi dengan teliti manakala analisis dinamik adalah analisis yang mengenal pasti perisian perosak melalui pemantauan. Walaupun kedua-dua analisis telah dilakukan, namun penambahbaikan harus dilakukan bagi mengesan perisian perosak dengan lebih tepat. Dengan zaman teknologi terkini, terlalu banyak cara yang boleh digunakan bagi penyerang untuk menyebar perisian perosak ke telefon pintar terutama pengguna Android. Dengan itu, kajian in telah mencadangkan sistem pengesan perisian perosak dengan menggunakan teknik pembelajaran mesin. Objektif kajian ini adalah untuk mengesan perisian perosak yang telah menyerang sistem operasi Android dengan lebih tepat. Hasil kajian yang sangat baik dapat membuktikan bahawa sistem pengesanan perisian perosak dapat mengesan perisian perosak Android dengan baik.

ABSTRACT

During this past year up until now, the total of malware that targets the Android operating system has increased compared to other operating systems. Therefore, an application that used the android operating system must be analyzed to detect the malware before the malware causes serious damage. To analyze the application, two types of analysis can be used which are static analysis and dynamic analysis. Static analysis is an analysis that is done by reviewing the codes diligently while dynamic analysis is an analysis that detects malware through observation. Even if both of the analysis was done successfully, however, an improvement needs to be done to detect the malware more accurately. With the technology arising now, there are many ways for the attacker to send malware to an Android smartphone user. Therefore, this research proposes a malware detection system using machine learning. This research objective is to detect the malware that attacks Android smartphones more accurately. The result of this research proves that the malware detection system is able to detect Android malware accurately.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

LIST OF ABBREVIATIONS

SVM	Support Vector Machine
KNN	K-Nearest Neighbour
RF	Random Forest
MLP	Multi-layer Perceptron
GUI	Graphical User Interface
VS	Visual Studio
CSV	Comma-separated values

CHAPTER 1

INTRODUCTION

1.1 Introduction

According to Android statistics as of 2022, the number of active android users around the world has managed to reach an all-time high of 2.5 billion users across 190 countries[1]. As the number of smartphone users increases, it is safe to assume that almost everyone in the world owns a smartphone. It is no wonder smartphones or mobile devices are considered a necessity nowadays since the various functions and services assist and ease users with their daily lives. Storing personal information, file access through cloud services, and browsing the internet is one of the many features of a smartphone. With the power to access almost everything from the touch of a screen and the ever-increasing number of smartphone users, it is no surprise that many security concerns and threats will rise. Attackers from any background can spread malware and viruses that can cause malicious effects on a user's smartphone. The most seen effect of this attack is when the smartphone's system starts acting differently than normal. Some malware is also capable of sending fraudulent or fake messages that can hijack and compromise the user's details and bank account.

Approximately 5,520,908 mobile malware, adware, and riskware attacks were prevented in the second quarter of 2022[2]. Continue to increase in frequency, with a 42% global year-on-year increase in attacks[3]. The incident demonstrates that a viable technique for detecting malware on Android smartphones must be developed immediately.

1.2 Problem Statement

In the era of globalisation, smartphones have become a necessity due to the numerous services they provide, which include connecting individuals around the world. The majority of users utilise smartphones for regular tasks such as document sharing, internet banking, and message sharing. Even while the usage of a smartphone has advantages for its user, there are disadvantages associated with the services supplied to the user. For example, storing confidential information on smartphones might attract the attacker to use dirty techniques to retrieve users' private information.

Therefore, several existing studies have presented various methodologies and methods for detecting malware. However, an improvement can be made to Android malware detection to increase detection efficiency and accuracy. The improvement can be performed by using different method or technique where all of which comes with a different result, especially accuracy.

Besides, Android users continue to be deceived by false alarms on their smartphones. As an example, more than a million people have downloaded a fake Android app that was pretending to be WhatsApp. The app resembled WhatsApp in an obvious attempt to deceive users into believing they were downloading an update for the popular chat service[4]. This demonstrates that not every method and technique can provide complete security against malware for Android users. Therefore, an improvement should be made to increase the security of the Android smartphone.

1.3 Objectives

The objective of this research are:

i. To review the present issues related to the Android malware detection system.

ii. To develop an Android malware detection system utilising Machine Learning.

iii. To evaluate the accuracy of the Android malware detection system's capabilities.

1.4 Scope of project

The scope of this research:

i) Platform

- This system only supports Android packages.

ii) Development / Functionality

- The system is only capable of detecting malware, not eliminating it from devices.
- The detection mechanism is only applicable to Android-powered smartphones.

iii) User

- Android smartphones users solely.

1.5 Thesis organization

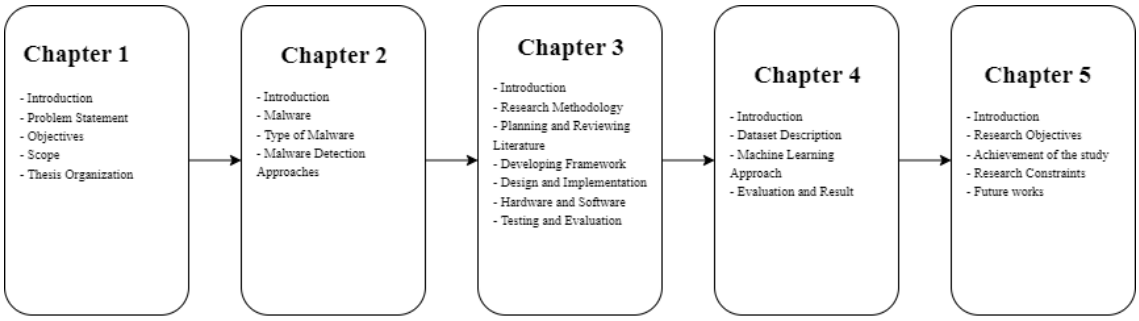

Figure 1.1 Overall chapter

Figure 1.1 shows the main chapters that will be elaborated including the introduction, literature review, and methodology.

Chapter 1, contains the introduction for the whole research and elaboration on the current issue. This includes the problem statement, objective, scope, and significance.

Chapter 2, contains a discussion regarding the literature review of existing research. The discussion explains a definition of malicious software, types of malware attacks, and the comparison of the solution of the present methodologies with earlier studies on the subject.

Chapter 3, demonstrate the discussion about the methodology used during this study. In this topic, the study discusses the data collection, data normalization and the software that has been used in this experiment.

Chapter 4, consists of the introduction of the chapter. Then, a details description of the dataset used in the research. Next, an explanation of each of the machine learning approach results. Lastly, evaluation of the resulting gain from all of the machine learning approach used in this research.

Finally, chapter 5 contains a discussion about the research objectives and the achievement of the study. It also discusses the research constraint and the improvement that can be made for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter will provide a literature review that might aid comprehension of malware detection techniques and their characteristics. A description of how malware can be discovered will also be examined. Besides, there will also be a comprehensive comparison of previous research, each of which employed a different technique to detect malware. The comparison will be made to improve the current work.

2.2 Malware

Malicious software, often known as malware, is an infiltration programme meant to harm and destroy computer systems. Many types of malware can be found and each of them has the capability to steal user information, and abuse it. Each malware can be detected by observing its particular actions which are backdoors, rootkits, spyware, worm, and etc.

In reality, malware can corrupt the operating system or any application in the devices by bypassing the device's access control. This will lead to unwanted actions from the device’s user such as accessing sensitive data, increasing battery consumption, and many more. Malware is also defined by its malicious actions against the system requirements which includes observing the end user activity without authorization.

2.3 Type of malware attack

Various types of malware can damage Android mobile devices. One of the most dangerous malware is malware that infiltrates the end user's mobile devices via bogus software updates, fraudulent applications, and spam emails. This malware deceives the user into executing malware websites and applications. Users tend to be deceived by the malware because the malware has been designed to resemble an authentic website and user-friendly applications.

However, protecting smartphones from malware attacks has become a significant hurdle for the attacker. The fact that attacker employed covert approaches by concealing the malware in the code to prevent detecting the malware and extend the malware through static analysis. In reality, every malware creator has a different objective which uses a different method to approach their target. Therefore, an exceptional method is a must to detect and prevent malware from spreading. Figure 2.1 shows the types of malware with the difference that is used by the attacker

Malware Type	Description
Rootkits	• Provide the hacker access to control over a target device. It is hard to detect because malware activities are executed while the user is not using their device.
Spyware	• Malware software that is used to disclose user private information through eavesdropping.
Botnet	• Able to infiltrate almost any device that is connected to the internet like a zombie.
Backdoor	• Neutralize normal authentication procedures in accessing the remote system to take control of the infected system.
Trojan Horse	• Malware conceals itself by pretending to be an ordinary application which attracts a user to execute the program. It

	has the ability to control resources and attack the accessibility of the operating system with denial of service.
Adware	• Malware software pops up an advertisement to deceive users to install malicious software on their devices.

Table 2.1 Type of malware

2.4 Malware Detection Approaches

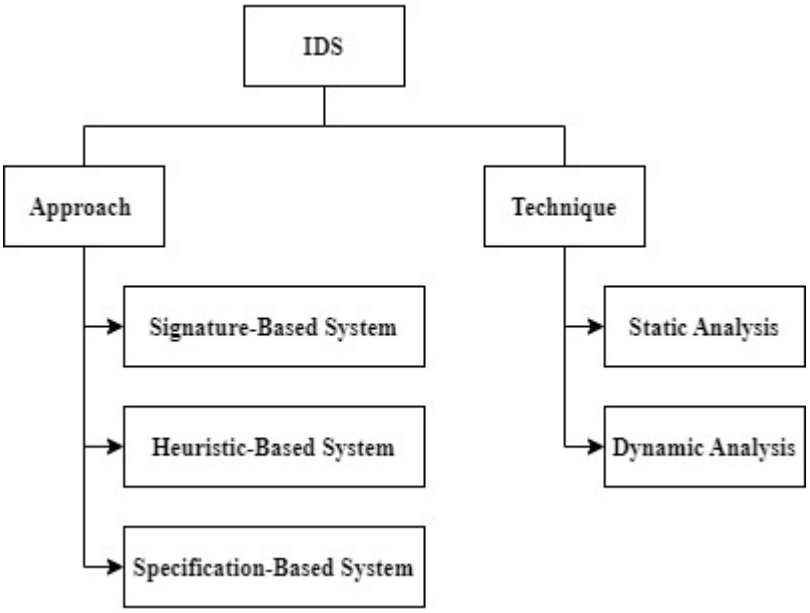

Figure 2.1 Malware Detection Approach

2.4.1 Specification-based Detection

Specification-based detection is virtually identical to anomaly detection which identifies a high false alarm rate. Instead of using machine learning to detect malware, it depends on the manual evolution of standards that control legal system behaviour. Consequently, this strategy relies on the programme specifications that apply to the objective of critical security programme behaviour. This method involved monitoring programme execution and finding deviations from the specification, despite detecting the circumstances of a particular assault. In addition, it estimated the application or system needs as opposed to attempting to design an application or system without precision.

A benefit of this strategy is that malware may be detected rapidly in both familiar and unexpected situations. However, it takes more time to create a detailed specification.

2.4.2 Signature-Based Detection

This type of detection, also known as abuse detection, must maintain a signature-based system and match the pattern to the database in order to detect malware. In other words, signature-based recognition also utilised the sample taken from different types of malware. This strategy is considerably more productive and efficient than others. Due to the fact that this signature was established by analysing the binary disassembly code of the infection, some antivirus programmes refer to this method. In addition, this disassembled code's characteristics were analysed and extracted.

In addition, these recovered attributes were used to build the malware family's signature. Since anti-virus software developers are continually updating and changing their unknown code, an approach based on signatures can detect malware more precisely.

However, there are still flaws that need to be investigated, rendering it incapable of identifying new or unknown viruses. This is because the document lacks a signature. Although it has a downside, it also has advantages. By deploying signature-based detection, malware can be precisely recognised while decreasing the number of resources required to detect malware and concentrating on signature-based attacks.

2.4.3 Heuristic-Based Detection

Heuristic detection, also known as anomaly-based detection, analyses the behaviour of both known and undiscovered malware. The behaviour parameter includes source and destination addresses, attachment types, and other statistically measurable information. There are two steps to this detection: the training phase and the detecting phase. During the training phase, the system's behaviour will be observed to assess the existence of an attack, whereas the goal of machine learning is to generate a profile of normal activity. On the other hand, the discrepancy between the detection phase of a usual

behaviour profile and the current behaviour will be classified as a potential assault. These components will be employed for behaviour detection:

i. Data Collection

- Use to gather both dynamic and static data.

ii. Interpreter

- The module for information accumulation will transform the raw information data into intermediate representations.

iii. Matching Algorithm

- The representation will be compared against the behaviour signature.

This method has the ability to detect new and unknown malware. Consequently, this method for identifying malware will focus on indicators of system behaviour. Moreover, this method demands an upgrade to the standard profile data describing statistics and system activities. Nonetheless, this strategy requires more disc space, RAM, and a high risk of false alarms.

2.4.4 Comparison Malware Detection Approaches

In order to identify malware, numerous detection types have been developed and their functionality enhanced. In Table 2.2, the differences between specification-based, signature-based, and heuristic malware detection methods are elaborated.

Malware Detection	Advantages	Disadvantage
Specification-Based	Any malware, whether known or unfamiliar, can be easily identified.	Takes longer to develop a specific specification
Signature-Based	Malware can be identified precisely while reducing the number of resources necessary to detect malware	Cannot detect unknown or new malware that lacks a signature.

	and concentrating on signature-based attacks.	
Heuristic-Based	Able to identify new and unidentified malware	Needs an update to the describing data of statistics and system behaviour in normal profile.

Table 2.2 Comparison of malware detection approaches

2.5 Analysis Techniques

There are two types of analytic techniques for identifying malware: static analysis and dynamic analysis. Malware can be analysed without execution using static analysis. In contrast, dynamic analysis involves the execution of malware prior to analysis. Both can be utilised to protect devices from malicious software (malware).

2.5.1 Static Analysis

Code analysis is another name for static analysis. Static will analyse the programme by examining it. For instance, the malware's source code will be analysed to gain an understanding of its operation. This technique employs a dissembler tool, debugger, and source code analyser, such as Interactive Disassembler (IDA), to build an assembly language source code from machine-executable code, which is then translated into an installation code that humans can comprehend. This opcode will be extracted as a feature to statically assess the application behaviour to detect the malware.

2.5.2 Dynamic Analysis

Behaviour analysis is another name for dynamic analysis. In this method, infected files are analysed in a simulated environment, such as the simulator and virtual machine, prior to identifying the file's general behaviour using SysAnalyzer, RegShot, and any other relevant tools. The file will then be detected upon execution in the real environment. During this execution, the system's interaction, behaviour, and machine output will also be observed. This strategy makes it easier to discover unknown viruses, but it requires a

great deal more work. This is because these techniques require extensive time to prepare the environment for malware analysis, such as sandboxes.

2.5.3 Comparison of Analysis Techniques

In order to detect malware, static analysis and dynamic analysis both have their advantages and limitations. Table 2.3 illustrates the distinction between static and dynamic analysis.

Type of Analysis	Description
Static Analysis	- Capable of identifying a novel or unfamiliar threat or malware. - Capable of analysing polymorphic and obfuscated threats and malware. - High accuracy - More secure and fast
Dynamic Analysis	- Incapable of identifying a new or unfamiliar threat or malware. - Incapable of analysing polymorphic and obfuscated threats and malware. - Low accuracy - Less secure and slow

Table 2.3 Comparison between static analysis and dynamic analysis

2.6 Machine Learning

Machine learning is an application of artificial intelligence (AI) that gives systems the ability to automatically learn and enhance their experience without being explicitly

programmed. Machine learning focuses on the creation of data-accessible computer programmes.

The learning process begins with observations of data, such as direct experience or teaching, in order to search for patterns in data and make future decisions based on the examples supplied. There are several techniques for machine learning:

Machine learning algorithms are often categorized as supervised or unsupervised algorithms.

- Supervised learning is capable of predicting outcomes accurately by using its labelled dataset to train the algorithm[5]. It uses a training dataset from the dataset to train the models in order to produce expected outcomes. The training dataset consists of inputs and real outputs. The dataset will be used to train the model over time. After sufficient training, the algorithm is able to predict the expected result based on the new input. The algorithm is also able to identify the output with the expected or real output, which will find errors to modify the model to minimize the error.
- Unsupervised learning is used to analyse and cluster unlabelled datasets[6]. This algorithm can identify hidden patterns or data clusters without the assistance of a human. It is one of the ideal solutions for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition because of its capacity to identify similarities and differences in information[6].
- Semi-supervised learning uses a sizable amount of unlabelled and a small amount of labelled data to train a predictive model. It works for a variety of problems, including clustering, association, and regression as well as classification[7]. This strategy can significantly increase the accuracy of learning systems.
- Reinforcement learning is a technique for teaching a machine to learn based on rewarding desired actions and/or undesirable ones. A reinforcement learning agent can perceive and understand its surrounding, act, and learn from mistakes[8].

Machine learning can be used for analysing a massive amount of data. Even though it produces more accurate results in identifying dangerous risks, it required longer time and resources to train the model. The power of processing a massive amount of data by using machine learning may be increased by integrating it with cognitive technologies and AI.

2.7 Related Review on Related Research

Machine learning has been used in much previous research in detecting malware on android. Each of the research has its method on how to solve the problem by using machine learning. However, every method has its limitations that could affect the accuracy of android malware detection. In this section, three existing research will be explained regarding the method used to detect malware on android.

2.7.1 Exploring Deep Reinforcement Learning for Android Malware Detection

In this research, the main purpose is to deploy reinforcement learning for cyber security issues and compare it to the existing work done using Deep Learning[9]. The number of the defected android device has increased compared to other OS platforms due to the third-party policy of Android. With all of the available malware detection techniques created by the past researcher, deep learning also has been used for detecting malware on Android. The dataset used in this research is the Drebin Dataset which is provided by the Technische Universitat Braunschweig. The dataset includes over 215 attributes and its feature derived. The dataset is extracted from 15,036 applications which consist of 5,560 malware files and 9,476 benign files. It is collected from August 2010 to October 2012. The training dataset that will be used in training the model is 75% of the total sample which is 11,277. The testing dataset is over 25% which is 3,759 samples for testing the model.

The type of machine learning used is reinforcement learning which is defined by a tuple of state, reward, and action. The algorithm used is Q learning which will decide the best solution of action based on the current state of the agent. The classifier used in this research is Random Forest Classifier and Extra Trees Classifier. By using Q-learning with Random Forest, the accuracy produced is 87.652% with a learning rate of 0.00039. Compared to Random Forest, Extra Trees Classifiers have produced more accuracy with 94.30% with the same learning rate of 0.00039.

In conclusion, Q-learning with Extra Trees Classifier produce the best accuracy of 94.30 using the sample dataset from Debrin Dataset compared to Random Forest. It is proof that by using reinforcement learning with the Q-learning algorithm, it can segregate the sample as either benign or malware. However, further exploration can be done toward the modelling for better accuracy.

2.7.2 Malware Detection: A framework for Reverse Engineered Android Applications through Machine Learning Algorithms

In this research, the author uses machine learning and reverse-engineered technique to detect android malware[10]. The major purpose of this research is to identify the most helpful criteria for detecting malware on Android. The researcher provides a novel subset of features for the static detection of Android malware. The subset consists of seven additional features from around 56000 features. The result obtained from detecting android malware is over 96.4% in accuracy with 0.3% false positive. To increase the prediction rate, the researcher implemented a Boosting ensemble learning approach with Decision Tree based on binary classification. The model is trained by using the latest sample of android malware collected in recent years. The algorithms used in this research are Naïve Bayes, Decision Tree, K-Neighbor, Gaussian NB, AdaBoost, Support Vector Machine, and Random forest. The best accuracy of all the machine learning algorithms is AdaBoost with an accuracy of 96.24%. The model does not achieve the highest peak in accuracy or predictive but it contributes by providing enhanced feature sets with the latest API level applications datasets.

In conclusion, the researcher has improved a framework that could detect Android malware. This proves that reverse engineering with the proposed method is able to detect the malware with an accuracy of 96.24% which is the highest accuracy in this research. However, the suggested approach has a limitation in terms of a lack of sustainability concerns and static analysis. The model resilience will be reconsidered in terms of enhanced and dynamic features.

2.7.3 Android Malware Detection Using Machine Learning with Feature Selection Based on the Genetic Algorithm

This research discusses the efficiency of applying machine learning feature selection with a genetic algorithm[11]. In the wider category of evolutionary algorithms, a genetic algorithm is a metaheuristic that draws inspiration from the process of natural selection. The algorithm cannot guarantee the best solution because the solution does not progress which results in the solution being poorly optimized. However, it could help find optimal combinations and find a solution that is hard to accomplish. The algorithms that are used in this research are Decision Tree, Random Forest, Decision Table, Naïve Bayes, MLP, SVM, Logistic Regression, AdaBoost, and K-NN. Each of the algorithms will be measured by using a confusion matrix. Based on the final result, the best accuracy is using SVM with an accuracy of 99.20%. The performance of the Naïve Bayes algorithm has the lowest accuracy compared to other algorithms which is 71.00%.

In conclusion, genetic algorithm-based feature selection is useful compared to common feature selection. It is able to drastically reduce the model's building time over non-selection. Even though this research needs to be improved, it is confirmed that genetic algorithms can help detect Android malware using machine learning.

Table 2.4 show the comparison of the three existing research in term of analysis and accuracy.

Research	1	2	3
Technique	Deep Reinforcement Learning	Reverse Engineered	Genetic Algorithm
Analysis	A training approach that emphasises rewarding good actions while penalising undesirable ones	Codes from decompiled mobile applications are automatically analysed.	A metaheuristic that draws inspiration from the process of natural selection.
Accuracy	94.30%	96.24%	99.20%

Table 2.4 Comparison between previous research 1, 2, and 3

2.7.4 Improvement from the existing system

The existing studies of malware detection systems utilised a dataset, features, precision, etc. to determine whether malware exists in the system or not. Despite this, none of the existing machine learning systems can prevent malware from entering a system. Therefore, the purpose of this research is to enhance an existing system by granting permission to dangerous code-free applications. In contrast, malware-containing software would be denied system access. By utilising both static and dynamic analysis, the system can detect malware.

2.8 Conclusion

This chapter is one of the essential components of this research, as can be concluded. This chapter compares previous solutions proposed by another researcher with Machine Learning, the current solution. In addition, this chapter describes the various approaches that assisted the researcher in achieving their proposed malware detection technique. So many techniques have been proposed over the past few decades. However, those strategies needed refining in order to have a better results in the future. Therefore, chapter 3 proposes the current method to assist Android users in identifying malware on their devices.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In the previous chapter, the explanation regarding malware has been explored including their restriction capabilities. Three existing research related to android malware detection also has been described briefly in chapter 2. This chapter will describe the proposed technique and approach that will be implemented during this research including the methodology of this research.

3.2 Research-Based

Figure 3.1 depicts the four key steps of a research-based approach, which include literature review, development of new architecture, design, and implementation, as well as testing and assessment. Research-based is appropriate to be used as the methodology in this research since the phase can be continuously examined to get accurate results. This research-based system development life cycle differs from those previously proposed. This is due to the fact that this strategy will emphasise handling and observing every detail of this topic's investigation.

There are 4 phases involved in the research-based approach. The first phase is planning and reviewing the literature. The previous research based on the research topic will be reviewed and analysed. After that, the research requirement will be defined including the problem statement, scope, and objective of the research. The second phase is developing the architecture. During this phase, a detailed analysis of previous research will be considered to observe the appropriate algorithm and method that will be used in this research. After the development of the architecture is completed, the design and

implementation of this research will be conducted. There are two requirements that will be needed for this research which are software and hardware. After all the requirement is prepared and ready to be used, the implementation of the design and detection model will be included. After the implementation is completely executed, the modal need to be tested and evaluated to observe the research constraint and improvement for further research.

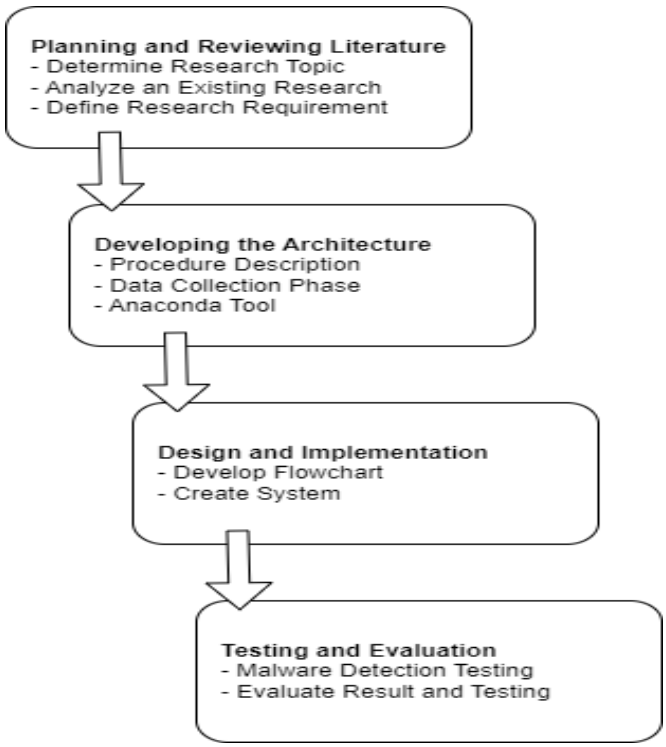

Figure 3.1 Main Phase of Research-Based

This research is modified because it is possible to revert to prior phases with little losses in order to incorporate the enhancements in the new research. Even more so, research-based methods permit occasional modifications to solve problems during the current phase. In addition, research-based methodologies allow researchers to quickly make adjustments to the requirement of the research project.

3.3 Planning and Reviewing Literature

The key phases of this research-based are planning and a relevant literature review. Before examining current studies, the relevant sort of research question is determined through conceptualization. When a research topic is selected, relevant publications, studies, and journals are compiled for study. Existing studies' analysis facilitates comprehension of the research subject. These allow for the clarification of the problem statement, the purpose, and the scope of this investigation. Using the suggested n-gram opcode, associated information on malicious code detection and their current code detection must be found for this research.

This research's sources consist of prior student references and Internet publications or articles. In addition, the current research is scrutinised and screened for relevance to the study question. Additionally, the collected material should be pertinent to research and used in the advancement of research.

The objective of research on various approaches and techniques is to determine which strategy and methodology are most suitable for resolving the problems that have occurred on Android devices, particularly those linked to malware. Since security is the primary concern for Android devices, this research must concentrate on malware detection for Android smartphones. Existing research studies on malware detection are evaluated critically and categorised according to the location where the malicious code structure was developed. Each proposed method of malware detection is analysed to determine its limits and contributions. Therefore, this type of information is essential for determining the testing approach employed by the researchers. Therefore, the constraints of previous research will be avoided in this research.

3.4 Developing the Architecture

On the basis of a previously examined strategy, it has been determined that Machine Learning will be employed to build malware detection. Figure 3.2 depicts the architecture development of the malware detection system.

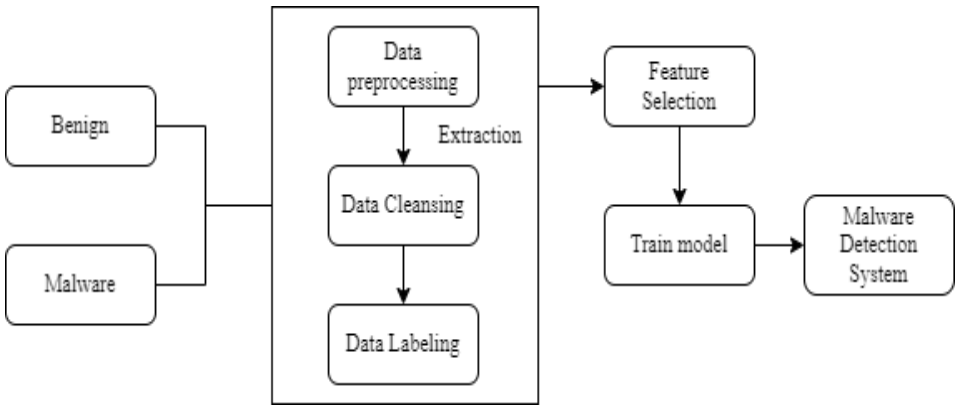

Figure 3.2 Malware Detection System Architecture

3.4.1 Procedure Description

The malware detection system architecture was used in developing a machine learning technique that can train a model by using dataset samples to learn the behaviour of benign and malware applications. This architecture is able to identify the new application status whether it is benign or malware. Furthermore, the architecture is a basic step in implementing machine learning algorithms and it cannot skip even a step. If one of the steps is skipped, the algorithm cannot be executed and can cause an error.

The data collection begins by gathering permissions which include the feature with the value of benign and malware. The data processing involved in this process is decompiling an apk file to extract the permissions. The data cleansing is where the dataset is observed which involves exploratory data analysis including descriptive analytics and handling missing values. Next, for the data labelling, all the permission gained from the extraction will be stored in an appropriate format and saved as either attribute-relation file format (x.arff) file or comma-separated value (CSV) before using them in Jupiter notebook.

Next, from the dataset that has been processed, a feature selection process will be conducted to find the best feature for model training. The purpose of this feature optimization technique is to differentiate the non-bio-inspired with a bio-inspired algorithm.

Furthermore, the data collection and feature selection process is the most crucial part of implementing a machine learning algorithm. The reason is that preparing a usable

dataset that is suitable for the model to be trained is difficult and sensitive which could affect the machine learning algorithm result. Then, this process will notify the database. From this moment, the filtering data will rely on the permission and its package names to ensure that the database is free of duplicate features and applications. Hence, the filtered features will be sent to the machine learning process for the optimization features.

3.4.2 Data Collection Phase

In the data collection phase, the process of data collection about benign and malware application datasets will be discussed. The sample dataset used in this research is drawn from the Kaggle website which was uploaded by Shashwat Timari[12]. The dataset consists of a feature vector of 215 attributes extracted from 15,036. There are 5,560 malware apps from the Drebin project and 9,476 benign apps. The table below shows the number of malware and benign application from the dataset.

Dataset	Source	Total Use in Experiment
Benign	Benign App	9476
Malware	Drebin	5560
Total		15036

Table 3.1 Dataset Summary

3.4.3 Decompiling the apk file

Firstly, the dataset of benign and malware applications is collected with a total of 15,036 samples with 9476 benign applications and 5560 malware applications. The malware application was collected from the Drebin dataset. The figure below shows the process of data collection.

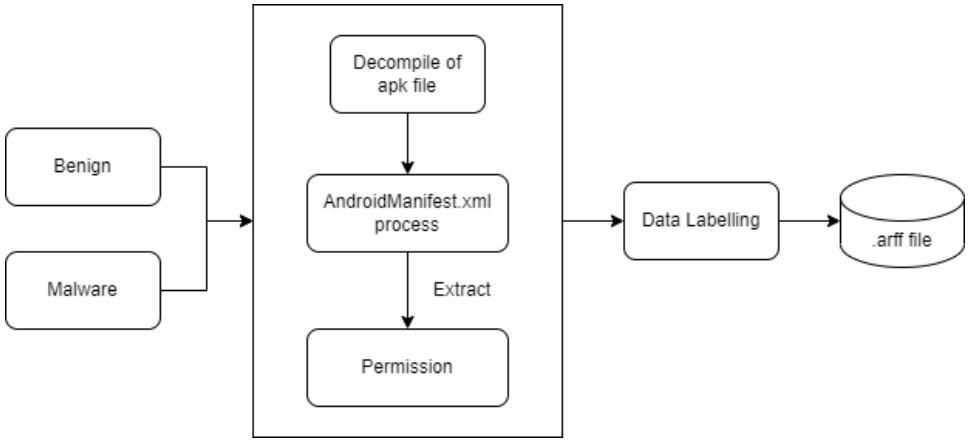

Figure 3.3 Data collection

The AdroidManifest.xml file is used to retrieve data required for this research such as android permission and activities. All of the extracted permission will be labelled before storing them in the database as an x.arff file or CSV file. All of the permission value stored is in binary number which is 0 or 1 except for class which stored either B(Benign) or S(Malware). Furthermore, feature optimization helps to find the best feature(permission) to be used. The table below shows the best permission from benign and malware applications.

Permission	Frequency
INTERNET	13111
READ_PHONE_STATE	9509
GET_ACCOUNTS	4492
SEND_SMS	3558
RECEIVE_SMS	2814
READ_SMS	2808
CAMERA	2054
WRITE_SMS	1702
MANAGE_ACCOUNTS	1565

USE_CREDENTIALS	1520
READ_HISTORY_BOOKMARKS	1397
RECORD_AUDIO	1357
READ_SYNC_SETTINGS	1252
RESTART_PACKAGES	1199
WRITE_HISTORY_BOOKMARKS	1193
WRITE_SYNC_SETTINGS	1159
INSTALL_PACKAGES	1052
AUTHENTICATE_ACCOUNTS	968
ACCESS_LOCATION_EXTRA_COMMANDS	779
WRITE_APN_SETTINGS	714

Table 3.2 Top twenty permissions from benign and malware applications

Based on the table above, there are three major permissions that have been requested in either benign or malware applications which are INTERNET, READ_PHONE_STATE, and GET_ACCOUNTS. INTERNET is the most frequent permission request which is 13111 requests. The other permissions also have a high value of frequency which need to be concerned. This shows that this is the android trend based on permission results nowadays.

3.4.4 Machine Learning Classifier

Machine learning is a subpart of Artificial Intelligence(AI) which was used in prediction, classification, and automation. It can be learned without the need of using explicit programming. It can predict future results and make better decisions when receiving new data. The process is called learning where the system will learn about the dataset and find the pattern for each result. The process and learning method are different for every classifier type which also give different result such as accuracy. The technique is widely used to classify samples in the intrusion detection system either benign or

malware. In this research, a supervised learning approach will be applied because the dataset has labels (malware and benign). Other than that, supervised learning is able to produce satisfying results via error reduction. In this research, four different classifiers will be used to observe the comparison of results between the classifiers. The classifiers are Random Forest (RF), K-Nearest Neighbour(KNN), Multi-Layer Perceptron(MLP), and Support Vector Machine(SVM).

Random Forest (RF): Popular collective learning technique for supervised classification and regression. This technique works by structuring a random set of decision tree and producing the class based on mean prediction (regression) or class category (classification)[13].

K-Nearest neighbour(KNN): In KNN, it depends on the nearest each data point. Each of the data points plays its role when determining the classes of new input data. The data will calculate the distance between the existing data point and the new data point. The nearest distance will be chosen. It can conclude that the new data point will be in the same classes as the chosen data point classes.

MLP: Multi-layer perceptron is a model of an artificial neural network. Multiple node layers made up the MLP, which interacted through weighted connections.

SVM: In Support Vector Machine, it has a data point, hyperplane, and margin. The data point is the point where each data will be placed at. The hyperplane is a line that separates a set of an object having different classes. A margin is a gap between the line and the data point. The distance between the line and the data point will be calculated. A good margin is a margin that has a larger distance between the line and the data point. To predict the value, all the data point located under the line is considered one category while the upper is considered another category[14].

3.4.5 Machine Learning Tool

Many machine learning tools can be used to execute the research. The purpose of machine learning tools is to run the analysis model development that produces data analysis. By training the model, it helps the system to learn the pattern or make a prediction by using a past dataset or present dataset. By implementing the machine learning tools, it provides the user with a feature that could help speed up the analytical

work. Other than that, it is also able to execute complex mathematical calculations to solve a problem without requiring expertise in machine learning. The machine learning tool that will be used in this research is Anaconda.

3.4.5.1 Anaconda

Anaconda is a data science platform that could run python and R programming languages and it is open-source software. At the current moment, there are around 30 million users from 236 countries and regions that use anaconda to perform complex data analysis[15]. Anaconda support many application and feature including cmd.exe prompt, Jupiter notebook, JupiterLab, VS Code, and etc. In this research, Jupiter notebook will be used to write the code and execute the model. By developing the system on Jupiter notebook, the researcher can add and edit the system anytime. Since it is a notebook, the outcome from each note can be displayed separately. This will make it easier for the researcher to observe the result for each note. It also can print out a report of the notebook in pdf format. The figure below displays the Anaconda Navigator desktop graphical user interface(GUI).

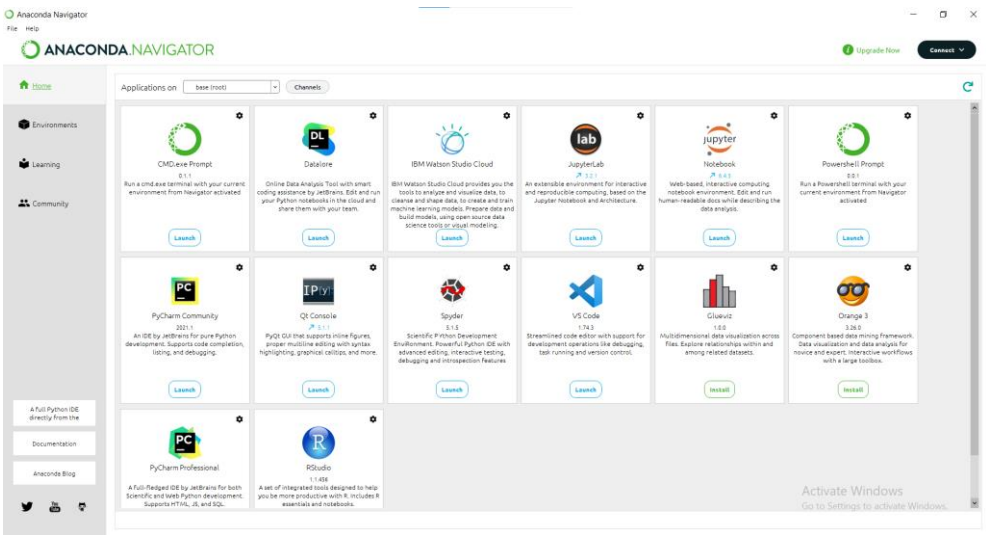

Figure 3.4 Anaconda Navigator

Anaconda Navigator consists of 4 main navigator buttons:

a. Home: display a list of default applications with the main application. All of the applications can be opened via the home interface without executing the program

manually. In this research, Jupiter notebook will be used to execute the machine learning algorithm.

b. Environment: contain a list of environments that will be used to execute the anaconda navigator. The user can create, import, clone, backup, or remove the environment.

c. Learning: Anaconda Navigator provides a learning platform for the user to learn about different topics such as python and Jupiter documentation.

d. Community: the user can communicate or engage in a community that has been prepared in the platform. The community is divided into two sections which are forum and social.

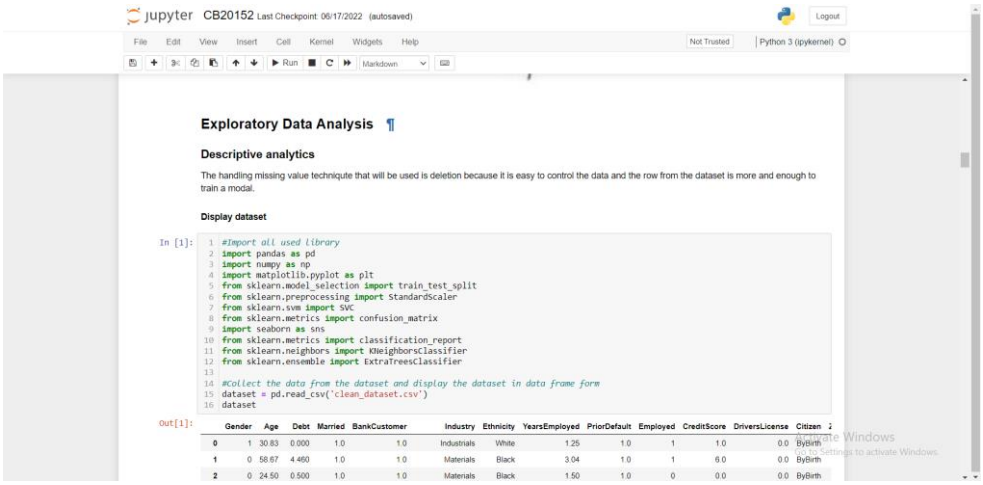

Figure 3.5 Jupiter notebook application

The figure above shows the example of the Jupiter notebook interface that was open in google chrome. The user must have knowledge of python language to use the application since the outcome are displayed through code. The user will need to import a library such as the sklearn library to execute each classifier in this application. This application is a good start for those who want to learn more about data science.

3.5 Design and Implementation

After the research framework has been developed, the framework needs to be approved. To test the accuracy of the malware detection, a draft is made before executing the implementation of the system. The figure below shows the flowchart of the procedure

to test the concept before moving forward to the malware detection system for Android smartphones.

There are 5 components of the design model which are exploratory data, defining a database, creating a system, testing, and evaluating the result. Each of the components will be briefly discussed in the next subtopic.

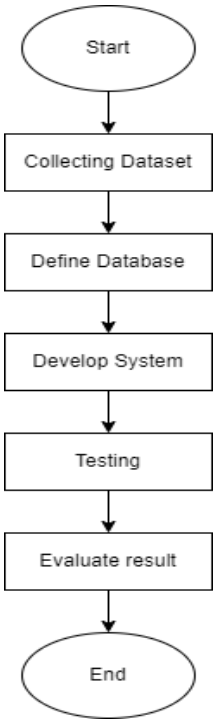

Figure 3.6 The flowchart for the malware detection method testing improvement

The final phase in this research is implementation. The implementation of the proposed solution will use the design model as a guideline during this phase. Then, software preparation such as Anaconda is an important step that needs to be considered during this research.

Next, the dataset that consists of an Android permission dataset will be explored in the system. Then, the research flow will follow according to the procedure which has been done during the designing phase.

3.6 Hardware and Software

In this research, there are hardware and software needed in order to train and test the model. The hardware and software requirements needed for the training and testing phase need to be specified in the early stages to carry out this research. All of the requirements needed will be used throughout this research in the next chapter.

3.6.1 Hardware Requirement

Hardware	Description
Processor: Intel® Core(TM) i5-7200U @ 2.50 GHz 2.70 GHz RAM: 8.00 GB System type: x64-based processor	Laptop specification that was used during documentation, testing, gathering references, and implementation for the entire research.

Table 3.3 Hardware requirement

3.6.2 Software Specification

Software	Description
Window 10	An operating system of the device used.
Anaconda	To execute machine learning algorithm and display results.
Microsoft Words 2016	To document the whole research documentation.
Microsoft Excel 2016	To create a Gantt chart and view the dataset.
Google Chrome	To search and download material.

Mendeley	To make citations for the research document.

Table 3.4 Software requirement

3.7 Testing and Evaluation

The final phase in this research is testing and evaluation. The system will be tested and the algorithm result needs to be evaluated. By conducting the testing and evaluation, it could help detect the best result and determine the limits of the system which can avoid the previous existing research constraint. The main objective of this research is to prove that the proposed solution is the best malware detection system with accurate results is claimed in this research. Hence, this phase is to identify the constraint and defects of this research that could be used to make an improvement to get the desired result. Furthermore, a thorough analysis will be made based on the experiment's results. Then, the research's hypothesis will next be evaluated to see if it has been proven or not.

Lastly, a clear explanation describing the entire process of this research is elaborated. The result is evaluated and recorded to view the achievement of this research. A full explanation regarding the implementation phase will be discussed in the next chapter.

3.8 Comparison results from each classifiers

By using the dataset in this research, the classifier can be executed and each of them produce different results. Table below shows that Random Forest have the highest accuracy compared to other classifiers. This shows that by using Random Forest classifier, the malware detection system can detect the malware more efficiently. For MLP and SVM does not have accuracy at this moment.

Classifiers	Accuracy
Random Forest	90.46%
K-Nearest Neighbour	90.31%

Multi-layer Perceptron	90.30%
Support Vector Machine	90.06%

Table 3.5 Comparison accuracy between classifiers

3.9 Conclusion

This chapter consists of preparation that the researcher will be used to conduct this research such as algorithm, research framework, and etc. The hardware and software requirements needed also been mentioned to execute the experiment and to gain the result. Next, a list of classifiers and methods that will be used are briefly explained because each of the classifiers has different usage. Then, the explanation of the process that will be executed is described such as testing and evaluation. In the next chapter, a detailed explanation regarding the implementation, testing, and evaluation will be further discussed.

CHAPTER 4

IMPLEMENTATION, RESULT AND DISCUSSION

4.1 Introduction

This chapter will provide the implementation from the methodology included in chapter 3. The implementation stage is very important and crucial during the development as it could affect the results. The result will be analysis and visualize to conclude the outcome from the process. Through the implementation, a discussion regarding the malware detection results will be made.

4.2 Dataset Description

It is important to decide which dataset that will be used in the implementation because it could affect the accuracy of the results. Exploratory data analysis is the early process in the implementation. The collected dataset will be explored and analysed to know the details explanation and understanding about the malware and benign activities. The dataset will be separated into training and testing data to see the result details and prediction.

The dataset titled “Android Malware Dataset for Machine Learning” was provided by Shashwat Timari at Kaggle website[12]. The dataset used in this research consists of 15,036 instances of 5,560 malware applications and 9,476 benign applications. The malware application in the dataset was collected from Drebin dataset. The label in the dataset named as “class” to identify whether the instance is benign application or malware application. The class value stored in short case alphabet which are “s” for malware and “b” for benign. The dataset consists of 3 types of features which are Manifest Permission (53%), API call signature (33%) and Others (14%). Most of the

feature used in this research is based on Manifest Permission feature which is declaration of the specific permission that an app requires to access the system resources or data.

4.3 Machine Learning Approach

By using machine learning approach, the machine is surely able to detect malware on an Android smartphone through Android permission features approach. This approach takes shorter training and testing time in detecting malware.

The dataset consists of many feature that can be used to detect the malware. However, not all of them are significant to detect malware. To find the most significant feature for malware detection, features selection process is included. Feature selection helps in identifying the most important feature that have the most impact on the label and removing irrelevant or redundant features that may not contribute much to the model performance. It also helps to reduce the complexity of the model and reducing time taken and cost of data acquisition, preparation and analysis. Table 4.1 shows list of features used in this research.

Permission	Description
INTERNET	Allow an application to access to the internet connection.
READ_PHONE_STATE	Allow an application to have access to the phone state of the device.
GET_ACCOUNTS	Allow an application to access
SEND_SMS	Allow an application to have access in sending SMS
RECEIVE_SMS	Allow an application to have access in receiving SMS

READ_SMS	Allow an application to have access in reading SMS
CAMERA	Allow an application to have access to camera
WRITE_SMS	Allow an application to have access in writing SMS
MANAGE_ACCOUNTS	Allow an application to have access in managing user accounts on the device.
USER_CREDENTIALS	Allow an application to have access the user's credentials
READ_HISTORY_BOOKMARKS	Allow an application to have access in reading history bookmarks
RECORD_AUDIO	Allow an application to have access in recording audio
READ_SYNC_SETTINGS	Allow an application to have access in reading sync settings
RESTART_PACKAGES	Allow an application to have access in restarting packages
WRITE_HISTORY_BOOKMARKS	Allow an application to have access in writing history bookmarks
WRITE_SYNC_SETTINGS	Allow an application to have access in writing sync settings
INSTALL_PACKAGES	Allow an application to have access in installing packages
AUTHENTICATE_ACCOUNTS	Allow an application to have access in authenticating accounts
ACCESS_LOCATION_EXTRA_COMMANDS	Allow an application to have access in location extra commands

WRITE_APN_SETTINGS	Allow an application to have access in writing APN settings

Table 4.1 List of permission features

4.4 Evaluation and results

The results show below are the outcome that obtained from five machine learning classifiers which are Random Forest, K-Nearest neighbour, Multi-layer perceptron, and Support Vector Machine. In this research, the result consists of five parameters that will be used to compare and analysis between each classifier. The parameters are accuracy, FP rate, precision, recall, and f-measure. Table 4.2 shows the outcome from testing set for the four selected classifiers.

Classifiers	Accuracy	FP rate	Precision	Recall	F-measure
Random Forest	90.46	1.48	96.84	76.89	85.72
K-Nearest neighbor (KNN)	90.31	1.58	96.62	76.65	85.48
Multi-layer perceptron (MLP)	90.30	1.66	96.48	76.83	85.54
Support Vector Machine	90.06	1.71	96.01	75.15	84.31

Table 4.2 Performance from each classifiers

The results from figure 4.2 shows that Random Forest has achieve the highest accuracy which is 90.42% compare to KNN which only 90.31%. From the comparison between classifier, Random Forest classifier are able to detect malware more effective compare to other classifier. Feature selection also contribute greatly in determining the

effectiveness of the Android malware detection. Based on the results, the precision rate affects the classifier in producing relevant and accurate result.

4.4.1 Confusion Matrix

A confusion matrix is a table that is used to evaluate the performance of a machine learning model by comparing the predicted values with the actual values. It is also known as an error matrix. The table below shows possible classes of prediction between benign and malware based on testing set. Table 4.3 shows the performance for each classifiers.

Classifier	Actual	Prediction	
		Benign	Malware
Random Forest	Benign	2789	42
	Malware	388	1291
K-Nearest Neighbour (KNN)	Benign	2786	45
	Malware	392	1287
Multi-layer Perceptron (MLP)	Benign	2594	44
	Malware	364	1207
Support Vector Machine	Benign	2858	50
	Malware	398	1204

Table 4.3 Confusion Matrix for each classifier

The table above shows that Random Forest prediction in identify malware and benign are better compare to other classifiers. The Random Forest produces correct results by predicting the malware with 128. In term of incorrect predicted, Random Forest predict the most minimal value for incorrect benign which is 42 and MLP predict the

most minimal value for incorrect malware which is 364. Based on the observation, Random Forest classifiers able to predict malware more accurate compare to other classifiers.

4.4.2 Receiver and operating characteristics curve (ROC)

The research classified processes as either malware or benign based on the Android manifest features. In addition to using performance metrics. Receiver Operating Characteristics (ROC) curve is calculated for each machine learning classifier. In this analysis, the True Positive Rate (TPR) is consider as the detection rate for correctly predicting malware processes, while the False Positive Rate (FPR) was selected as the detection rate for incorrectly predicting normal processes as malware.

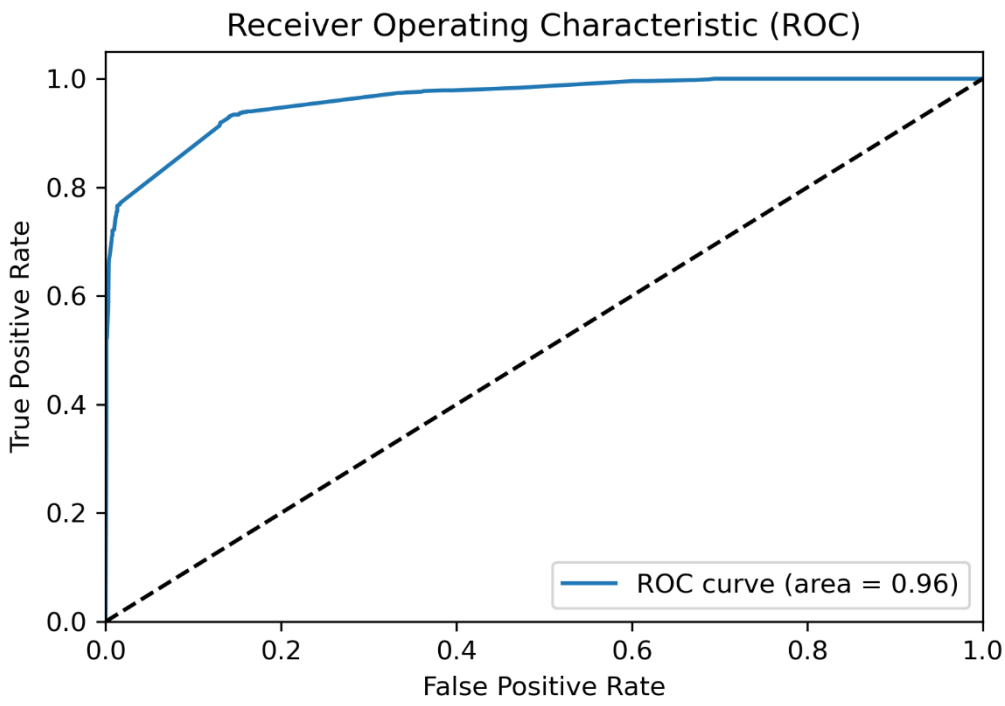

Figure 4.1 ROC cure for Random Forest

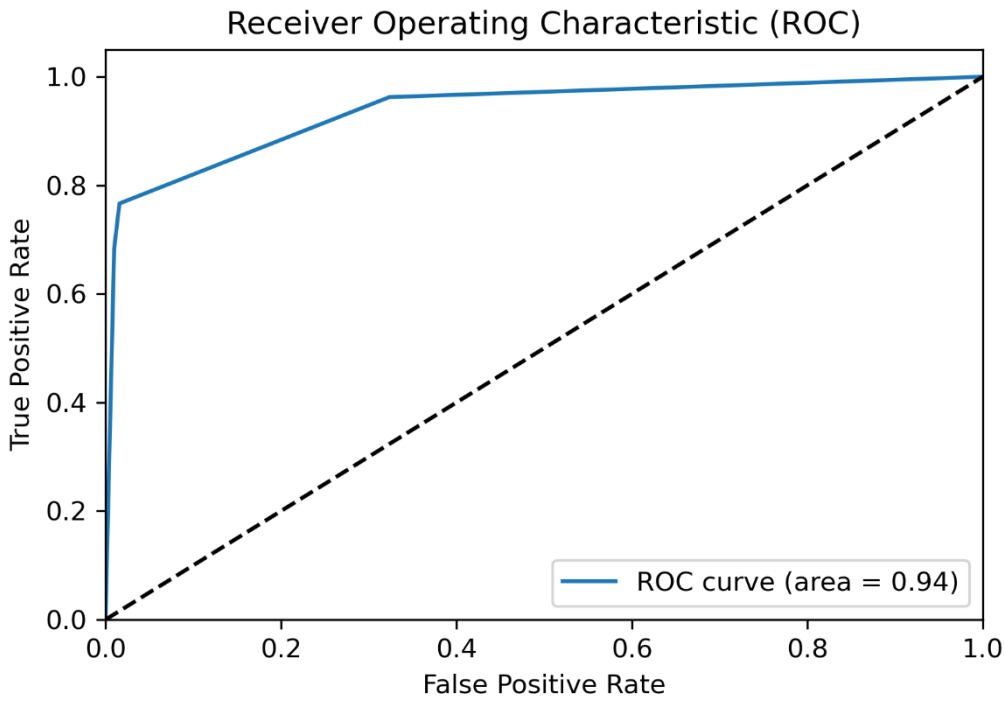

Figure 4.2 ROC curve for KNN

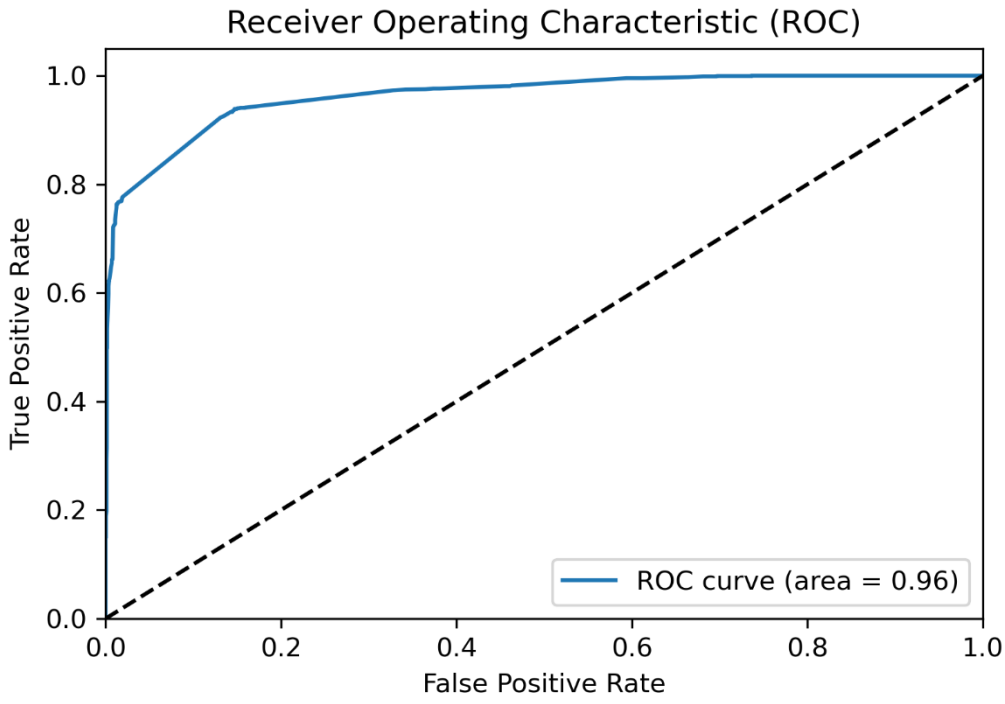

Figure 4.3 ROC curve for MLP

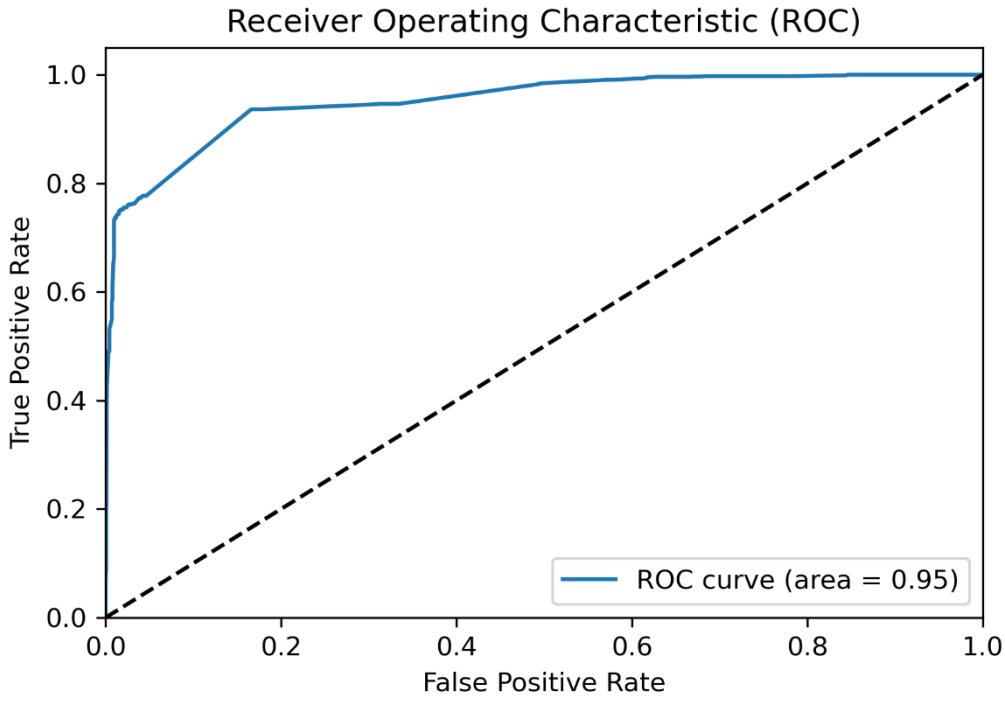

Figure 4.4 ROC curve for SVM

The x-axis in figure 4.1, 4.2, 4.3, and 4.4 represents the rate of error detection, while the y-axis represents the detection rate. Comparing the ROC curves directly is challenging due to the similar appearance when examined under the same conditions. To address this, the area under the curve (AUC) will be used to measure the detection accuracy. The AUC of 1 indicates perfect prediction, while AUC of 0.5 indicates poor prediction. Table 4.4 present the performance results based on AUC measurements.

Classifier	AUC	Prediction
Random Forest	0.96	Perfect Prediction
K-Nearest Neighbor	0.94	Perfect Prediction
Multi-Layer Perceptron	0.96	Perfect Prediction
Support Vector Machine	0.95	Perfect Prediction

Table 4.4 AUC result

CHAPTER 5

CONCLUSION

5.1 Introduction

In this day and age, Android smartphone have become a useful device that have changed human life style. With the increasing of smartphone users, many applications have been developed that help utilising the smartphone usage. There is various type of application that have been created for its Android user such as educational, games, social networking, and shopping applications. Some application need the user personal information to utilise the application feature smoothly. For example, a shopping application need the user address and phone number to send parcel and contact the user for any inquiry. Downloading an application on Android smartphone have become easier nowadays as it can be downloaded anytime and anywhere as long as the smartphone connect to the internet. Internet have become an important service because most of the applications require internet connection. With the internet connection, a smartphone has become the best medium for communication and data sharing through application. In spite of the benefits from using the applications, it also has the disadvantages in term of security and integrity. Most people did not understand the importance of security and integrity of the smartphone and the harm that could occur to them.

This research provides the understanding of malware application and malware detection by using machine learning. The dataset used in this research consist of android manifest permission and feature selection approach has been performed to list out the most relevant permissions. The feature selection process has decrease the size of dataset and the number of permission used. The machine learning classifiers that involve in this research are Random Forest, Multi-Layer Perceptron, K-Nearest Neighbor, and Support Vector Machines. The parameters are taken into consideration to detect malware application effectively.

5.2 Research Objectives Revisit

The purpose of this research is to improve a malware detection system by using machine learning. There are three research objectives of this research that was listed in chapter 1.

The first objective is to review the present issues related to the Android malware detection system. This objective was achieved at chapter 2 where three research thesis that publish in online scholarly journals was carefully reviewed and compared. In addition, chapter 2 consist of information regarding malware detection system and the classification of malware detection and machine learning approach.

The second objective is to develop an Android malware detection system utilising Machine Learning. The objective was achieved at chapter 3 and 4 which involve in the model implementation and result. The evaluation regarding Android malware detection system was evaluated through Jupyter Notebook Anaconda software. All the testing executed will produce five result that will be used to evaluate the model: accuracy, False Positive Rate (FPR), precision, recall, and f-measure.

The third objective is to evaluate the accuracy of the Android malware detection system's capabilities. This objective was achieved in chapter 4 where all the model result has been written and the performance was evaluated. Based on the model result, Random Forest have the highest percentage of accuracy in detecting Android malware.

5.3 Achievement of the study

This research started by observing the evolution of machine learning system through literature review. The past thesis reviewed the detection of malware application issues and feature selection. Four machine learning classifier have executed and tested while the result had been evaluated. The result from each classifier was evaluated which fulfil the objectives of this research.

5.3.1 A detection model for malware

This research has developed a model that can detect malware application based on Android permission by means of a dynamic analysis. An approach of machine learning has been used as it is a better adaptive detection model. All of the model are able to detect malware on android very well based using the given dataset.

5.3.2 Issues in Android malware detection studies

In chapter 2, malware detection type and its relevance in malware detection was presented. There were three past research thesis that was used to compare and review. By drawing out each of the past research thesis strengths and weaknesses, a great understanding and analysis could be performed. To improve the effectiveness of the malware detection system, an observation has been carried out to identify the best classifier and dataset that will be used as the limitation of the past thesis is known. An addition to it, relevant features could be listed to increase the efficiency of the detection system.

5.3.3 Issues in Android malware permission selection

This research shows that feature selection is very important in improving detection performance and minimizing its complexity. The selected feature will affect the model result as the feature importance is not the same for every feature in the dataset. In addition, the number of selected feature will affect the accuracy of the model and not all of the feature need to be involved in the execution.

5.4 Research Constraints

It is confirmed that the research satisfactorily achieved its aim and objectives from the discussion in the previous chapter. In spite of the achievement, there are several number of constraints and limitation found in the research which will be mentioned for the future reference.

5.4.1 Sample Size

The sample size used in this research is 15,036. It is difficult to identify the significant relationships of the data. A large sample size is required in this research as it will impact the result and to ensure a representative distribution of the populations. A small number of data tend to cause imbalanced data and model over fitting or under fitting due to the small data scale, and too high or low feature dimensions[16].

5.4.2 Time

The time given to complete the whole research including investigating research problem, exploratory data analysis, executing machine learning model, and manage changes or stability are limited with the due date of the given task.

5.5 Future works

There are three recommendations for future work that could be made for further improvement. The following future work outside the scope of this research are listed as follows:

5.5.1 Enhance false alarm rate

False alarm rate involves in any of the machine learning model. It will occur when the model incorrectly identifies a sample class. This means that the model incorrectly predicted a sample from malware class as belong to the benign class. It will affect the accuracy of the malware detection which can cause huge impact in term of result. Therefore, a reliable and effective technique to enhance the false alarm rate is needed to improve the model's ability to minimize false alarms and increase its reliability and usefulness.

5.5.2 Selection of relevant features

Complex dataset is difficult to handle especially in feature selection process to improve the model detection performances. It requires further analysis to identify the correlation between benign and malware applications. By performing further analysis, this will reduce false which will increase the detection accuracy.

5.5.3 Deep data analysis and result evaluation

This research can perform deep data analysis and result evaluation. A deep data analysis would be able to provide a clear information about the dataset description. It can display the faulty or any anomaly in the dataset before performing data cleaning. In addition, the model should include other result that could contribute to the result evaluation such as time taken for the module to finish execute. The time taken can be used in comparing the time taken for each of the classifier to determine the best classifier to detect malware.

REFERENCES

Use a reference manager such as *Mendeley* or *EndNote* to generate your list of references here.

[1] DAVID CURRY, "Android Statistics (2022)," *Business of Apps*, 2022. https://www.businessofapps.com/data/android-statistics/#:~:text=Android is the most popular,users spanning over 190 countries. (accessed Nov. 29, 2022).

[2] T. SHISHKOVA, "IT threat evolution in Q2 2022. Mobile statistics | Securelist," *Securitylist*, 2022. https://securelist.com/it-threat-evolution-in-q2-2022-mobile-statistics/107123/ (accessed Dec. 01, 2022).

[3] "The Mobile Malware Landscape in 2022 - Check Point Software," *Check Point*, 2022. https://blog.checkpoint.com/2022/09/15/the-mobile-malware-landscape-in-2022-of-spyware-zero-click-attacks-smishing-and-store-security/ (accessed Dec. 02, 2022).

[4] Lorenzo Franceschi-Bicchierai, "More Than 1 Million People Downloaded a Fake WhatsApp Android App," *Vice*, 2017. https://www.vice.com/en/article/evbakk/fake-whatsapp-android-app-1-million-downloads (accessed Dec. 02, 2022).

[5] "What is Supervised Learning? | IBM." https://www.ibm.com/my-en/topics/supervised-learning (accessed Jan. 17, 2023).

[6] "What is Unsupervised Learning? | IBM." https://www.ibm.com/my-en/topics/unsupervised-learning (accessed Jan. 17, 2023).

[7] "Semi-Supervised Learning, Explained | AltexSoft." https://www.altexsoft.com/blog/semi-supervised-learning/ (accessed Jan. 17, 2023).

[8] "What is Reinforcement Learning? A Comprehensive Overview." https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning (accessed Jan. 17, 2023).

[9] S. Gill *et al.*, "Exploring Deep Reinforcement Learning for Android Malware Detection," *EasyChair Prepr. no. 6594*, vol. 2, no. July, pp. 1–8, 2021, [Online]. Available: https://easychair.org/publications/preprint/MhtN.

[10] B. Urooj, M. A. Shah, C. Maple, M. K. Abbasi, and S. Riasat, "Malware Detection: A Framework for Reverse Engineered Android Applications Through Machine Learning Algorithms," *IEEE Access*, vol. 10, pp. 89031–89050, 2022, doi: 10.1109/ACCESS.2022.3149053.

[11] J. Lee, H. Jang, S. Ha, and Y. Yoon, "Android malware detection using machine learning with feature selection based on the genetic algorithm," *Mathematics*, vol. 9, no.

21, pp. 1–20, 2021, doi: 10.3390/math9212813.

[12] "Android Malware Dataset for Machine Learning | Kaggle." https://www.kaggle.com/datasets/shashwatwork/android-malware-dataset-for-machine-learning?select=drebin-215-dataset-5560malware-9476-benign.csv (accessed Jan. 22, 2023).

[13] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random Forests and Decision Trees," no. December 2013, 2012.

[14] T. Evgeniou and M. Pontil, "Support Vector Machines : Theory and Applications WORKSHOP ON SUPPORT VECTOR MACHINES : THEORY AND APPLICATIONS," no. May, 2014, doi: 10.1007/3-540-44673-7.

[15] "Anaconda (Python distribution) - Wikipedia." https://en.wikipedia.org/wiki/Anaconda_(Python_distribution) (accessed Jan. 22, 2023).

[16] P. Xu, X. Ji, M. Li, and W. Lu, "Small data machine learning in materials science," *npj Comput. Mater. 2023 91*, vol. 9, no. 1, pp. 1–15, Mar. 2023, doi: 10.1038/s41524-023-01000-z.

APPENDIX A
SAMPLE APPENDIX 1

APPENDIX B
SAMPLE APPENDIX 2