OPTIMISED SUPPORT VECTOR MACHINE (SVM) FOR DETECTION OF ANDROID MALWARE WITH NEIGHBOURHOOD COMPONENT ANALYSIS ALGORITHM

BY

EFEFIONG INYANG UDO-NYA

MTECH/SICT/2017/7617

DEPARTMENT OF CYBER SECURITY SCIENCE

FEDERAL UNIVERSITY OF TECHNOLOGY

MINNA

SEPTEMBER 2021

OPTIMISED SUPPORT VECTOR MACHINE(SVM) FOR DETECTION OF ANDROID MALWARE WITH NEIGHBOURHOOD COMPONENT ANALYSIS ALGORITHM

BY

EFEFIONG INYANG UDO-NYA

MTECH/SICT/2017/7617

A THESIS SUBMITTED TO THE POSTGRADUATE SCHOOL FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA, NIGERIA IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF MASTER OF TECHNOLOGY IN CYBER SECURITY SCIENCE

SEPTEMBER 2021

ABSTRACT

The exponential surge in android malware has continued to increase with new variants of the malware surfacing daily. This is due to the ubiquitous nature of mobile platforms and internet technology which are almost inevitable in today's digital age. One of the approaches used in detecting malware is the use of machine learning algorithm. SVM is a machine learning (ML) classifier that has demonstrated promising strength to be used as a tool for the detection of android malware. The goal of this study was to build an optimised SVM model for detection of android malware using the neighbourhood component analysis (NCA) algorithm. The research work adopted the neighbourhood component analysis (NCA) algorithm to sieve out irrelevant features which guaranteed excellent model performance. The Bayesian optimization method was used to optimally combine the various SVM hyperparameters and their values to have a super and optimised NCA-BOM-SVM model for the detection of android malware. The results of the proposed model (NCA-SVM) show an accuracy of 97.8%, false alarm rate of 0.021, precision of 97.9%, error rate of 0.02, recall of 97.9%, and f1_score of 97.9%. These results show an enhanced performance of malware classification with higher accuracy and precision, alongside reduced false alarm rate and error rate, thus demonstrating an improvement on existing literature. This implies that the Bayesian optimization of SVM alongside the neighbourhood component analysis algorithm provides a more robust tool for the detection of android malware.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

1.0 INTRODUCTION

1.1 Background to the Study

Smartphones remain one of the most popular technologies in high demand, due to their ubiquitous nature as a result of their adaptable functionalities and diverse usage. The use of mobile phones has spread across different facets and spheres of life and it is almost indispensable in the present technological, modern and digital world. The functionalities of mobile platforms have widened, with an increasing scope of acceptance which tends to outpace laptops and personal computers. In fact, there is no gainsaying that the advent and sophistication of mobile technology has enhanced better efficiency and effectiveness of human life and activity.

The sophistication and increased functionalities of mobile platform have not just increased their complexities, vulnerabilities and risks, but have drawn attention of hackers and cybercriminals, culminating in the violation of users' privacy in barrage of ways, and in various cyberattacks. Smartphones have become primary target of hacking activities, with various devastating or degree of loss, as some malware developers catch in on these vulnerabilities to gain unauthorised access and privilege thus posing as security threats. Suffice it to say that cutting-edge technology via Internet of things (IoT), and proliferation of information technology (IT) devices and infrastructures like smart phones, computers, tablets, has created the leeway and a quite conducive environment and platform for creation and proliferation of malware. As such, research in the security of smartphone technology has become an issue of great and global concern and its interest not just to the smartphone research community, but the entire tech-world.

Android is considered a leader with an overwhelming market share in the world of smartphones operating system (OS), having gained or acquired superb or very large market share on smartphones and tablets as billions of devices are used around the world. This can be attributed to the market openness, ease of operability, and easy access compared to other mobile OS. Research shows that android platform remains one of the most patronised or used mobile platform in the present digital age. Consequently, android applications have exponentially grown. It was reported that Google Play had more than 2.99 million apps as at 2020. Similarly, malicious apps are rapidly increasing as it was reported that almost two million new apps that are malicious, were noticed as at the third quarter of 2020.

Malwares are malicious software or applications designed to target OSs, computer systems, or network infrastructures for the purpose of financial gains, information or data disruption, destruction, or theft; distorting the normal workings of the system or probably grind them to a halt. Most of these malwares have specific targets which could be smartphones or specific mobile OS; network, banking systems, manufacturing systems like the stuxnet. As such, there are mobile malwares designed to target smartphones, while some are specifically designed to target mobile platforms in particular like the Android OS. Interestingly, software vulnerabilities on Android smartphones is undoubtedly on the increase and quite challenging to detect or identify, due to complexity of the smartphone platforms. This exploitation triggers unusual or unexpected behaviours in the system. Android mobile OS is reported as the most targeted and affected mobile OS by malware threats. Malicious applications leverage on opportunity from the openness of the Android platform to carry out attacks. Pertinent to note that hackers now attack android mobile platforms with a sophistication that beats the security mechanism of those devices. While anti-malware developers are creating algorithms to contain and avert existing malwares, malware developers on their part are re-strategizing and enhancing new techniques to help them perpetrate their nefarious activities, thus making these two sets of persons to constantly be at their heels to outsmart the other. It is expedient to note that despite the security mechanism embedded in android mobile platform and the security awareness of the users, the rate of malware attacks on android devices is still on the increase.

The security architecture of Android projects a primary line of defence which restricts applications from running outside an isolated environment alongside with restricted privileges to the apps from the permission system. Despite the isolation and restricted privileges, apps still exploit the system and kernel vulnerabilities to bypass the Android defence. This is affirmed by the tremendous threat cases over the past years. There is tremendous increase in the size or number of malwares with so much improvement in sophistication of attack. This motivates the unrelenting concern in the research on smartphone security.

Though several efforts have been made by several researchers, heralded with remarkable success, in stemming the activities, attacks, and effects of malwares on smartphones and other mobile targets, new or recent malware attack vectors are emerging every day and some have eluded these anti-malware detection systems. This remains the reason for continuous research in this problem domain. Researches to counter malware attacks using machine learning have been on-going. Thus, developing an efficient model or system that can robustly efficiently detect android malware is quite expedient, as it will offer protection to the device and further evade cyberattacks on android platform. Though Support Vector Machine (SVM) have been used in various researches to build models for android malware classification, this research work will focus on developing an optimised SVM for android malware classification, that will offer a more optimal performance than existing ones. SVM based models and frameworks have considered by researchers as being to deal with code location and applications that are malicious and to handle labels with multiclass (Wu *et. al.* (2021).

1.2 Statement of the Research Problem

The existing malware detection data models are not good enough (Wu *et. al*. (2021) in the selection of representative features, therefore a need to improve on the selection technique to achieve better feature selection. In addition, the existing android malware detection models are still subjected to unacceptable accuracy rate and false alarm rate such as developed by Rana *et. al*. (2018) and Li *et. al*. (2018**)**, and therefore need improvement.

1.3 Aim and Objectives of the Study

The aim of this research is to develop an optimised SVM model for detection of android malware with neighbourhood component analysis (NCA) algorithm.

The objectives of this research are to:

i. Develop data model to obtain representative features with Neighbourhood component analysis algorithm.
ii. Develop an optimised SVM model with Bayesian optimisation method for detection of android malware.
iii. Evaluate the performance of the model in (ii) using the relevant machine learning performance evaluation metrics

1.4 Scope of the Study

The focus of this research is to develop an optimised support vector machine model using the Bayesian optimization method, and train it with an optimised data features obtained from feature selection algorithms, which will consequently produce a model with optimal predicting capacity. More so, the research work will utilise the android malware and benign datasets from drebin data source. The technological tool for implementation, optimisation, and evaluation is the Matlab 9.6.0 platform.

1.5 Significance of the Study

The data model will ensure the elimination of the curse of dimensionality on the data consequent upon which there will be reduction in the features to a manageable and accurate data while the detection model will assist to detect android malware in applications. The evaluation will ensure better performance for the detection model. This will be of great benefit to the academic/research world alongside every other sector of life as the risks and threats associated with android malware will significantly reduce.

CHAPTER TWO

2.0 LITERATURE REVIEW

2.1 RELATED LITERATURE

Adebayo and Aziz (2014) carried out a static code analysis on android malware where they examined the different attack vectors on mobile android platforms for the purpose of identifying and obtaining useful features for further analysis and classification. They also researched into several attack vectors that target android operating system. This research, by analysing the Zertsecurity, highlighted the various step for statically analysing of android malware and came to conclusion that Zertsecurity is a Trojan that steals login information.

Adebayo and AbdulAziz (2014) proposed the classification of android malware using the Apriori algorithm with an improvement with the particle swarm optimization and static code analysis. They used the static code analysis to extract features from android applications bytecode which was used to train supervised classifier. The improved apriori algorithm was used as the selection technique. In the course of the research work, they carried out both supervised and unsupervised classification and the results maximised and minimised the true positive rate as well as the false positive rate respectively.

Khan *et. al.* (2017) researched and proposed a solution and method of malware classification and detection using the interceptor that sits in between the web browser and the server. The research engaged in the static analysis of JavaScript code for feature extraction. They further reduced the dimensionality of the datasets through wrapper method of feature selection where a sizeable feature subset was obtained and deployed for classification.

Dataset used for the research work had 1924 instances of 409 and 1515 malicious and benign JavaScript respectively. They had 3 different experiments with different levels of partitions. In experiment 1, the entire subset of the dataset after feature selection was used for the training, whereas, in experiment 2 and 3, the hold-out method with eighty percent to twenty percent; and 10-fold cross-validation respectively. The following machine learning classifiers were engaged in the research work – Naïve Bayes, KNN, svm, and J48. SVM in all the three experiments achieved accuracy of 94.55% and 95.42%, compared to other classifiers like Naïve Bayes which had 95.06% and 97.99%; J48 with 99.22% and 98.64%, in the second and third experiments respectively.

Khan *et. al.* (2017) proposed an intelligent framework that profiles android users, which use multi-layer technology to communicate with the control sever and monitor their mobile devices. This framework assists in detecting malware application and activities of botnet on those devices. The proposed system creates user profiles which will be used for behavioural analysis, by monitoring resources used by the devices and all incoming and outgoing text messages.

Wen and Yu (2017) extracted several Android features using both the static analysis and dynamic analysis methods. Considered the relief method for feature selection algorithm though with some weakness of inability to eliminate redundant features; alongside the PCA which is a dimensional redundant algorithm, capable of transforming features linearly into a low dimensional (or capable of reducing the dimensionality of the data). They proposed a new feature selection method called the PCA-RELEIF, which they used to dispose off redundant features and further carried out dimensionality reduction on the data. SVM was used to build a classifier. According to them, the experimental result obtained from the comparative analysis of using Relief, PCA, and PCA-RELIEF as feature selection methods with SVM classifier, shows that PCA-RELIEF produced the best accuracy, highest TPR, and the lowest FPR.

Yuan *et. al*. (2016) researched and proposed the DroidDectector, which depends on deep learning (DL) and also runs on the web, designed to detect malware on android devices. The DroidDetector produced a DL model which was thoroughly validated, tested and evaluated by performing in-depth analysis on features from real-world apps obtained from Google Play Store, and several malwares datasets from Contagio community and Genome projects. The researcher commended the strength of Deep Belief Network (DBN) in Android malware characterization and further reported and accuracy of 96.76%. They asserted that their accuracy rate significantly outperforms some ML techniques such as Multi-layer Perceptron, Logistic Regression and support vector machine.

Traditional malware detection method/techniques have gone obsolete due to the sophisticated nature of Android malware obfuscation and detection avoidance method, as these malwares continue outsmart several detection methods and further inflict harm to devices and system resources, Alzaylaee *et. al* (2020) proposed a deep learning solution called the DL-Droid which has the capacity through dynamic analysis to detect malware in Android applications. This study trained a deep learning algorithm which produced a DL model. Over 30,000 malicious and benign real-life applications on mobile devices were used to perform the experiments.

Rana *et. al*. (2018) proposed a string-based feature selection approach to remove irrelevant predictors and subsequently deployed the resulting dataset to train some machine learning algorithms which include decision tree, gradient tree boosting, random forest, extremely randomised tree algorithms. The dataset for the experiment was obtained from the Drebin dataset. Upon evaluation of these various classifiers, random forest classifier outperformed others with an accuracy of 97.24%, recall of 96.88%, fi_score of 97.23% and precision of 97.58%.

Rana *et. al.* (2018) evaluated different machine learning algorithms for the detection of malware on android devices. They achieved the purpose of their research by carrying out an analysis on drebin dataset. According to them, in order to ascertain what happened after decompiling Android Apk file and to check the permission and API functions, they took a swap at the AndroidManifest.xml file. The metrics evaluation on the classification on the static analysis showed random forest classifier outperforming others with an accuracy of 94.33% and recall of 94.27%. More so, the SVM had an accuracy of 90.74%, precision of 91%and recall of 91%.

Alzaylaee *et. al* (2020) researched and proposed DL-Droid, which is a framework for the detection of android malware. Their research carried out a comparative analysis involving dataset with dynamic features and one that has both dynamic and static features. According to their study, they achieved 97.8% detection for the former, and 99.6% detection rate for the latter. After extensive comparison of their results with seven popular traditional machine learning techniques, they asserted that their proposed system outperforms those ML classifiers.

Li *et. al*. (2016) researched and proposed the DroidDeepLearner which is a malware detection approach that is based weight adjustment. This weight adjusted approach which according to the research has the capacity to automatically detect and distinguish between malware and benign samples, deploys both the risky permission and API calls to build a Deep Belief Network model. The experiment utilised the dremin dataset with 237 features and the evaluation showed an accuracy of 90% accuracy.

Adebayo and Abdul Aziz (2019) proposed a novel solution for the detection of android malware with a knowledge-based database discovery model obtained from android benign and malware. These researchers gathered and extracted benign and malicious android applications after they had analysed the sample of the code. They utilised triple feature selection approaches for ranking the features in the order of importance further had an association rule based on the features. The combination of the parameters associated with apriori algorithm and the optimised generation of candidate detectors was used for feature selection. The detection of malicious android applications was made with extraction algorithm and rule models which was obtained from the candidate detectors from particle swarm optimisation with apriori association rule. The proposed method demonstrated a remarkable improvement over the existing contemporary android detection methods.

Chavan *et al* (2019) carried out a research on the comparative analysis on the classification of statically extracted features from Android applications. The research work covered both the binary classification and multi-class classification with family of Android malwares and their focus was on the permission requests by application. The research work used the Android malware Genome project dataset which consist of apk files which they obtained from various malware forums as well as Android applications. The benign dataset was obtained from the PlayDrone project. The researchers dealt with the challenge of high feature dimensionality using the information gain approach to reduce the features of the dataset and further used the RFE based on a linear SVM get the feature weights which was the criterion for feature elimination based on this approach. Subsequently, they deployed some machine learning classifiers (Adaboost, ANN, J48, LMT, linear SVM, random tree, and random forest on each of the resulting data subset from the information gain approach and the RFE with svm approach for malware classification and detection, alongside the cross-validation strategy. Evaluation and comparison of the created models was made with the precision and AUC. For information gain approach, the best precision value came from ANN and random tree with score of 0.97. The AUC for them in the information gained approach remain 0.94, 0.96, 0.96, 0.97, 0.97, and 0.99 respectively. Similarly, the precision for linear svm and J48 using the RFE with svm feature selection are 0.96 and 0.96.

Wang *et al* (2019) postulated the quality of datasets largely determines the dependency of malware detection models. This entails that some performances of some models may be unsatisfactory due to poor training datasets and can lead to failures of these models. They proposed SEdroid is an ensemble-based system with genetic algorithm. The SEdroid, which is an Android malware detection engine that is quite robust and engaging. They reported that SEdroid demonstrated 98.3% precision and 98.1% recall ratio. The research posits that designing SEdroid with consideration to diversity of the ensemble and accuracy, facilitate and fast-track the process of finding optimal ensemble combination, thus providing the model with super robustness and very strong generalization ability.

Yang *et. al* (2020) researched into means of improving on the accuracy and efficiency of Android malware classification/ detection and came up with an approach which is an ensemble of decision tree and support vector machine algorithm (DT-SVM). According to the researchers, this DT-SVM machine learning advanced algorithm which they designed extracted the Dalvik opcode of sample using the reversing Android software, the n-gram model was used to generate the eigenvectors of the sample. They trained the samples and consequently generated a decision tree. Subsequently, using the bottom up approach, the decision nodes were updated as SVM nodes. This research deployed the strength of both the DT and SVM especially overfitting reduction by SVMs, to have high accuracy. Their work achieved an all-time precision of 96% using the DT-SVM algorithm, for the Android malware apps classification/detection with a relatively low time consumption.

2.2 Android Architecture

Android was founded with the intention of developing the Android OS for mobile. It remained under the radar, until it was purchased by Google, Inc. in 2005. Android development incredibly soared higher as it captured almost 50% of mobile operating system market share. The 1.0 version of android OS was officially launched on September 23, 2008 and it ran on HTC Dream device. The Android OS experienced a rapid growth is attributed to one of its unique features of open source since the binaries source codes were released. This open source gesture makes it possible for design and building of mobile phones that runs on Android OS by any interacting person. Android OS takes approximately 2.6GB of disk space, as such the entire source code can be downloaded. Android left its Android Operating System as open source software until version 3.0 and above, and have remained closed source since then. The Android architecture has four main components

- ❖ The kernel
- ❖ The libraries and Dalvik machine
- ❖ The application framework
- ❖ The applications

2.2.1 The kernel

The kernel is one of the components of the Android architecture for its mobile platform that communicates and interfaces with the hardware device that it sits on. Alongside other functions, it takes care of device drivers, networking, security, process management as well as power, and memory management. It is accessible at http://android.git.kernel.org/ Application developers build applications in conformity with the Android kernel while the hardware or device manufacturer may have the leverage of kernel modification such that the Android OS is composite or works well with their particular hardware.

2.2.2 The libraries

The libraries component interfaces between the kernel and framework of the application as a translation layer. Developers can access these libraries which are written in C/C++ using the Java application framework via Java API.

Below are the core libraries though not limited to the list.

- ❖ LibWebCore; gives access to the web browser
- ❖ Media libraries – which makes it possible to access to audio and video
- ❖ Graphics libraries – which provides access to 2D and 3D graphics drawing engines

The Dalvik virtual machine is one of the runtime components that interact with and applications.

2.2.3 The dalvik virtual machine

The Dalvik Virtual Machine was designed and build to primarily make it possible for applications to execute in devices with very limited resources. This is quite typical of mobile phones. Virtual machine functions as a guest operating system that runs within another host operating system and operates by executing applications in a manner that portrays it as physically running in that machine. It is highly portable. This feature allows the developer to execute one code on any hardware platform that runs a compatible VM. “The Dalvik VM executes .dex files.

2.2.4 Android application framework

This layer is always known as the application programming interface (API) component, which make available to application developer a suite of services. It gives the developer access to user interface components, privileges of apps showing data between them via the common content providers, access to notification manager which alerts the device owners of events, and ability to manage the lifecycle of application through the activity manager in the framework. Application layer provides the space through which apps executes. This is the closest component that interfaces with the end user and it is place that the contact, phone, messaging, as well as the Angry Birds apps resides. There is a high tendency of mobile device users to lose their sensitive data or have the data mutilated or destroyed or have the privacy compromised, in addition to losing or having their devices stolen. The focus of the developer should be to develop applications with best functionalities as well as offer adequate protection for users' data.

2.3 Android Security Architecture

The Android security architecture through the Android kernel adopts or ensure the implementation of the privilege separation model during application execution. This entails that all running applications has its own user identifier as well as the group identifier. By this security feature, applications are forbidden from reading or writing to other applications or processes. More so, it also forbids applications from arbitrarily connecting to remote servers using the device's networking stack. Two applications running on sandboxes can only access the data of the other through an explicit request and permission granted accordingly. All applications that will eventually require to access other components of the systems must have this designed and built into the application by the developers. Android security architecture gives the end user the privilege of performing the final approval process and the prompt for permissions is expected to come at install time. The Android application code signing feature makes use of the certificate of individual developers in the identification and establishment of trust relationships amongst the different applications that runs in the Android OS. Android OS run only applications that has been signed with a self-signed certificate. Android security focuses on Android built-in security end permission and architecture. The Android platform is endowed with several security mechanisms which provides control over the security of the system and applications. This mechanism also implements the principle of application isolation as well as compartmentalizing every stage. Every process within Android has its own set of privileges. Except there is an explicit permission provided by the end user, other application can access this application or its data. APIs cannot be use without obtaining access from end user. Each process runs in its own isolated environment. Unless there is an explicit permission from end user, there is no interaction possible between applications. Interactions between applications is only made possible via permissions.

2.4 Machine Learning

Machine learning is a technique that provide systems' ability to autonomously make decisions from a set of provided data, without any external support. ML makes such decisions by first learning from the dataset and further understanding its patterns. "The big data giants like Google, Facebook, Amazon are using Machine Learning to gain maximum benefits from data and compete their rivalries". There exist various ML algorithms which are the support vector machine, decision tree, logistic regression, and random forest. The choice of

2.5 Supervised Learning

Supervised learning, is used for data modelling where there is a precise mapping between input and output data. The algorithm for supervised learning has the capacity to recognise and identify the relationships between the two variables in order to have a prediction for a new outcome. The following are supervised learning algorithms – Support Vector Machine (SVM), Gradient Bootstring, Artificial neural networks, Random Forest (RF), Linear Regression, and Logistic Regression amongst others. Classification is the process of recognising and grouping ideas and items into pre-defined categories or sub-classes. It is the process of deploying algorithm which use pattern recognition in the training dataset in order to spot the various patterns which could be number sequences, sentiments or similar words in future or new datasets. Classification is a type of supervised learning. In classification, algorithms make predictions on the likelihood of a subsequent dataset falling into predetermined categories, using input training data. This is made possible by the strength of an algorithm to analyse the sets of training data. Machine learning program deploy different algorithms to classify dataset into various categories using already categorised training datasets. Structured data can be classified as well as unstructured data. A supervised learning is mostly used for classification problems, considering the versatile features which help to actualise both categorical or its independent variable. Binary classification has two possible results or outcomes.

There are several types of classification algorithms and their usage depends on a dataset.

2.5.1 Logistic regression

Logistic regression gives an estimate discrete values which is based on given set of independent variables. It fits data into a logit function and consequently give the prediction of the probability of occurrence of an event. Regression values fall between 0 and 1. Steps like interaction terms, removing features, use of non-linear models and regularising of techniques can be deployed to improve the model. This is the analysis for independent variable to predict the binary outcome whose result falls into of the two categories. The dependent variable which is the outcome is always categorical while the independent variables can be categorical or numeric.

2.5.2 Decision trees

Decision tree is a machine learning algorithm that splits population of items into two or more homogeneous sets, taking cognisance of the most significant attributes which makes the group as distinct as possible. In decision tree classification, data points or sets are separated into two similar categories per time and such trickles down from the trunk to branches and to the leaves, where the categories are assumed to become more finitely alike.

2.5.3 Bayes classification

Naïve Bayes classification is a technique of classification, which is based on Bayes' theorem as well as independence between the samples or predictors. It portrays non-correlations or non-relations between features of the same class. Naïve Bayes has strength that can make it to outperform some sophisticated classifiers depending on the type of data set.

2.5.4 K-nearest neighbour (KNN)

KNN classifies new classes by searching for the k-nearest neighbour sample of the same category and attributes. It uses the various distance functions like Hamming, Manhattan, Minkwoski and Euclidean to calculate the distance of the k neighbour. While the three former distance functions are used for continuous variables, hamming distance function is used for categorical variable. At times, selecting the value of k might be a big challenge while performing KNN modelling. It is expedient to note that KNN is quite expensive with regards to computation. The variables are expected to be normalised to avoid higher range of values from being biased. More so, KNN provides optimal result when much work has been done at pre-processing stage like dealing with outliers and removing noise. KNN classifier uses the pattern recognition technique on a training dataset to find the k closest relatives in future datasets. It determines the place data within the category of its nearest neighbour.

2.5.5 Support vector machine (SVM)

SVM is a simple and robust classifier that works by creating a line which is the hyper plane that falls between two different sets of classes. This line is the classifier. This classifier classifies new data based on where the testing data falls which can be on either side of the line. SVM uses algorithm to train and classify data within decrees of polarity, taking it to a degree beyond X/Y predicts. It performs a non-linear classification by using the kernel to transform the data into higher dimension.

2.6 Malware Classification

Malware classification is the process that assigns a malware sample based on some factor into some specific malware families. Common attributes and properties are what malwares share in common and this makes it possible for the creation of signatures which could be used for their detection or classification. Malware within a family, shares similar properties that can be used to create signatures for detection and classification. Depending on the method of extraction, malware can either static or dynamic.

2.7 Types of Malware Analysis

Static and Dynamic analysis are two approaches for analysing malware files. The static analysis approach does not run the program, rather, it directly extracts features from the byte-code or disassembled instruction. It has an advantage of using less resources and ability to follow all possible execution path. However, this approach is quite sensible "to packing technologies, encryptions, compression, garbage code insertion, and code permutation", thus making it possible to bypass malware detection systems based on static analysis using obfuscation technique.

The Dynamics Analysis approach monitors the various behaviours of malware which include but not limited to tracking the flow of information, file system, process monitoring, instruction tracing, detection of system change, monitoring of registers, network monitoring, auto-start extensibility points, function parameter analysis, and the monitoring of function call. This approach is not sensitive to packing or obfuscation techniques. This implies that this approach cannot be bypassed by packing or obfuscation techniques, due to insensitivity to packing or obfuscation.

2.8 Malwares

Malware is a malicious software designed and implemented by hackers/attackers to meet the harmful or malicious intent or to carry out certain nefarious activities. The intent is to spread itself and remain undetectable, cause changes or damages, disrupt or gain unauthorised access to users' devices and inflict harm to data, infected system or network and or people in different ways. These nefarious activities range from fraudulent penetration of networks; compromise of computer and smart devices and bringing down devices' performance to knees; destruction and crippling of critical information systems and infrastructures, stealing of confidential information, amongst others Ransomware, rootkits, viruses, bots, spyware, Trojans, worms amongst others are some of the programs/malware designed or used by attackers.

Mobile malwares are specifically written to attack mobile platforms whose devices include smartphones, tablets, smartwatches and other wearable devices. It explores and exploits vulnerabilities of the mobile OS and phone technology It remains a growing threat to consumer devices. Malware remains one the biggest and toughest threats to mobile devices, information systems and in the internet at large. Each malware operates in a bewildering variety of forms with different attack vectors. There is a tremendously and significant increase in the varieties of mobile malware programs whose targets are smartphones and tablet, and the growth rate is highly alarming. The emergence of mobile malware experienced a significant explosion in 2011 at the reported of new incidents in the Android platform. Cybercriminals design malwares with the capacity to install themselves or are installed on the various devices by unwitting mobile users. The mode of distribution of malicious mobile programs are through the internet; downloads; and installation through device messaging functions.

The sophistication of malware attacks has increased as cybercriminals have turned to file-less now. This level of sophistication where a malicious code does not require an executable file in the endpoints, has made it more difficult and challenging for detection by the traditional antivirus (AV), due to low footprint as well as the absence of files to scan. Cybercriminals carry out these acts by injecting these malwares into some processes and execute only in the RAM. Detection in this case can only be achieved by studying the behaviours and malicious patterns of the processes. The world experiences mega breach of cybersecurity attacks which are always calibrated yearly based on the level of impact or devastation. In 2016, there was an alarming wave of wannacry ransomware attacks which attacked millions of computers across the globe. There exist different types of mobile malware variation with varying attack vectors, different methods of distribution and infection, and impacts on mobile devices.

In order to get optional protection of devices and business system, and further forestall or avert compromise of these systems, it is quite crucial and expedite for users to recognise the different types of malware and their operational procedures. According to IT security professionals, several generic, mobile-specific and other device-specific malwares have been designed by hackers/cybercriminals to prey on IT infrastructures' features and vulnerabilities like ones on smartphones and tablets.

2.8.1 Classification of mobile malware

2.8.1.1 *Worms*

Worms are malicious software which upon installing itself into the computer memory, replicate itself and infect the entire device or network. It spreads through software vulnerabilities or phishing attacks. Worms can perpetrate serious harm depending on the type of worm and possibly the security measures established on the device or network. It can modify and delete files; inject malicious software into IT devices or electronics infrastructures; replicate themselves severally to deplete and overwhelm the system resources; steal data; and install a convenient backdoor for hackers, amongst others. It spreads very fast, consumes network bandwidth, and overload as well as overwhelm a web server.

2.8.1.2 *Virus*

Virus is a malicious software which operates by attaching itself to an executable file. It needs an infected active OS or program to function and remains dormant, and can only be activated by launching the host file or program that it is attached to. It spreads to the entire system via this means. It can spread through websites, file sharing, email attachment downloads, and other downloads from unreliable websites. A computer or mobile virus can hijack applications, use these applications on the system to sneeze all over everywhere, by sending out files that are infected to other systems, clients, or friends

2.8.1.3 *Bots and botnets*

A bot is an IT device like computers or mobile device that is infected with malware such that it can be remotely controlled by cybercriminals and could be used to launch cyber-attacks. A collection of these bots also referred to as zombie, form a botnet which is limitless. Botnets can control millions of devices even as it continues to spread without being detected. Hackers through the master servant commands, use the botnets to carry out several malicious activities including sending spam and phishing messages; screenshots key logs and webcam access; and DDOS attacks. The sophistication of mobile malware has increased to the level that programs can operate secretly run without notice in the background on the user device and watching out for certain behaviours like online banking session.

2.8.1.4 *Trojan horses*

Trojan horse disguises itself as a real and trustworthy file or program. Mobile Trojan finds itself into devices by attaching itself to legitimate programs that does not look harmful and get installed alongside with the apps after which it will infect the device or perpetrate malicious actions. It is activated by users. Cybercriminals typically embed Trojans into files or apps in the mobile devices that does looks legitimate. The Trojan is activated by the user as they open a file and it can infect and deactivate other applications and the mobile device itself as soon as it is activated. It can also paralyse the device after a certain period of time or a certain number of operations. These malicious programs hijack the browser and captures user login details. Trojans themselves are a doorway. It can spy on devices or systems; capture or steal data; delete or modify data; harvest devices and make it part of botnet; and gain unauthorised access to devices and networks. Banking Trojans target vulnerable users by distributing fake version of legitimate mobile apps.

2.8.1.5 *Ransomware*

Ransomware is a malicious software that uses encryption principle to lock the victim's data on their device or locks the hardware devices, thus restricting devices or users access to their hardware devices, files or data with a demand for a payment of ransom which most times are with cryptocurrencies such as bitcoin, before the data or device is decrypted. In a ransomware attack, the victim is usually notified of an exploit on his device and instructions are further provided on how to recover the encrypted item, while the identity of the cybercriminal remains unknown. This makes it different from other attacks. In May 2017, a ransomware named Wanncry attacked and compromised thousands of computers across the globe. This attacked individuals and corporate bodies with monumental damages and losses.

2.8.1.6 *Adware and scams*

These are malwares that automatically deliver advertisements which serves pop-ups and display unsolicited or illegal ads which often does not have relevance to the users. Adware is often annoying, pose as nuisance to users and can slow down the devices. It can redirect or link users to malicious sites. A device that has it installed can deliver spywares, which most often are easily hacked, thus making the devices to be soft target for hackers, phishers, and scammers. Most adware is authored by advertising firms as a means of generating revenue. Though some adware is meant to only deliver ads, some of them to be bundled with spyware to track user activities as well as steal personal and confidential information.

2.8.1.7 *Spyware*

Malicious software with a common threat which secretly keep records of all activities of the users (both online and offline), harvest the users' data and collect personal and confidential information such as contacts, usernames, passwords, location, downloads, user preferences, messaging habits, browser history and surfing habits/ behaviour and relays these data to a third party. It can also collect device information like the IMSI number, product ID, IMEI number, and OS version, which can be used by the third party to launch future cyber-attacks. Spyware is often installed or distributed on user device without the user's consent as a freeware or shareware with a disguised or appealing function at the front end as a legitimate app, with covert, nefarious and unknown mission running in the background. This means is often use for perpetrating identity theft and credit card fraud. Spyware at times are referred to as adware because they may be advertisers or marketing firms. Cybercriminals or advertisers have access to users' data through spyware and some of them can further install additional malware that make changes to the settings of devices.

2.9 Feature Selection

Feature selection is a technique and ML process in which the dimensionality of data is reduced by sequentially selecting a subset of the input features that are most relevant to the predicted or target class (variable), to develop a model. Subject to some evaluation criterion or constraints which includes the size of the subset, required features, and excluded features, the feature selection algorithms search and select a subset of the features that are highly representative of the original dataset and will optimally model the measured responses. Despite the importance of irrelevant or redundant predictor or input variable, they can mislead and add more stress to the learning algorithm, which possibly can result in lower prediction performance. Though all features of the dataset can contain some level of information about the response class or variable, too many features will obviously degrade the model prediction performance. The main reasons or benefits for deploying feature selection algorithms in ML are to offer better or improve prediction performance with regards to accuracy and other evaluation metrics, facilitate or enhance faster and cost-effective predictions, and offer a better insight. These algorithms can be categorised into three types or methods

2.9.1 Filter feature selection

This method gives adequate measure of the importance of the features using the univariate statistics, entropy or correlation. The characteristics of the features like the variance and its relevance to the response class provide the basis for the measuring the feature importance. This method does not have any correlation with the training algorithm as important and relevant features based on the algorithm are selected as part of the data pre-processing step, and subsequently used to train a classifier.

2.9.2 Wrapper feature selection

This method selects a subset of the features and through training, sequentially adds or removes additional features with reference to a selection criterion. The removal and addition of features due to the change in model performance is as a result of the selection criterion which actually measures the change in model. Trainings and improvements are repeated by the algorithm and stops when its halting criteria is satisfied.

2.9.3 Embedded feature selection

In the embedded type feature selection method, the features importance is learnt as part of the model learning process. The importance of the features are obtained while training a model and the algorithm select the features that work well with a particular learning process.

2.10 Feature Selection Algorithms

Neighbourhood Component Analysis (NCA)- NCA is a non-parametric method which selects features with the hallmark goal of maximising prediction accuracy of regression and classification algorithms. The NCA feature selection with regularization learns features weights for minimization of an objective function that measures the average leave-one-out classification or regression loss over the training data. It learns the feature weights by using a diagonal adaptation with regularization. The feature selection is performed using the predictions and response variables. It learns feature weights for minimization of an objective function that measure the average leave-one-out classification loss over the training data.

2.10.1 Maximum relevance minimum redundancy (MRMR) algorithm

The MRMR algorithm's ultimate goal is to search for relevant set of features that are dissimilar and can represent the response variable effectively. It maximizes the relevance of a feature set and further minimizes the redundancy of a feature set to the response class. The MRMR uses the mutual information of variable-pairwise mutual information of features, alongside mutual information of the response class to quantify the redundancy and relevance of features.

2.11 Bayesian Optimization

Optimization is the process where a point is located that has the capacity to minimise the objective function which is a real-valued function known. The Bayesian optimization uses the Gaussian kernel function in the process model of the objective function, which is used for training the model. The acquisition function is a function use by Bayesian optimization to determine the next point to evaluate during optimization. The acquisition function as use in Bayesian optimisation explores areas and points that need to be remodelled and can also balance sampling at points that have low modelled objective functions. The Bayesian optimization algorithm seeks to minimize a scaler objective function in a bounded domain.

2.12 Research Gap from Literature

Table 2.1 provides a summary of related literature that were reviewed. This points to the fact that different researchers have carried out researches on the detection of Android malware using different methods and obtained various levels of performance results. Pertinent to mention that some of the performance recorded indicate insufficient accuracy rate and high error rates. More so, in some of the reviewed works, relevant metrics like accuracy rate, precision, error rate and false alarm rate were not reported. Despite these efforts, malware has continued to exponentially penetrate the mobile Android platform. Hence, the need for an enhanced and better performed machine learning algorithm that will provide better performance using the relevant evaluation metrics. This research work demonstrates strength in bridging these limitations using the Bayesian optimized SVM alongside the neighbourhood component analysis algorithm for detection of Android malware. All the relevant performance evaluation metrics which includes accuracy rate, false alarm rate, precision, error rate, recall, and f1_score are reported in this research work.

Table 2.1 Summary of Result Performance for Related Literature

Reference	Method	Accuracy (%)	False Alarm Rate (%)	Precision (%)	Recall (%)	F1_ Score (%)
Yuan *et al* (2016)	Deep Learning	96.76				
Khan *et al* (2017)	SVM	95.42	NR			
	NB	97.99	NR			
Rana *et al* (2018)	Random Forest SV	94.33	Not Reported (NR)	94	95	94
	DT	90.74		91	91	91
	ERT	91.78		88	94	91
	BAGG	93.66		93	93	93
		93.71		94	94	94
Rana *et al* (2018)	DT	96.13	NR	96	96	96
	RF	97.24		98	97	97
	Gradient Boosting	93.68		94	93	94
	Ext. Randomized	96.97		97	97	97
Li *et al* (2018)	Deep Learning	90	NR	NR	NR	NR
Wang *et a*l (2019)	Ensemble Learning			98.3	98.1	
Alzaylaee *et al* (2020)	Deep Learning	97.8				
Yang *et al* (2020)				96		

CHAPTER `THREE

3.0 RESEARCH METHODOLOGY

This section of the research work encapsulates and outlines in detail the various steps and procedures for achieving the aim and objectives of the proposed research work. The dataset was optimised to reduce the curse of dimensionality using the sequential feature selection (SFS) algorithm alongside three other feature selection algorithms. This optimised dataset was divided into training and test sets. The training set was used to train the SVM classifier, using the k-fold cross validation strategy to avoid overfitting.

An optimised SVM was tuned, trained and validated using the Bayesian optimisation method. The optimised, and the first SVM model alongside other classifiers were tested with the test dataset, and their predictive performance were evaluated using the standard evaluation metrics. The detailed description of the above procedures is given in the sections below.

3.1 Proposed Model

The proposed model involves collection of android applications (malicious and benign) for the design of the model. The features were extracted from the applications to form the n by n dimensional data vectors for the training and testing of data model. The features were optimised after the feature selection using the neighbourhood component analysis (NCA), maximum relevant minimum redundant (MRMR), and the sequential feature selection. The dataset with optimised features were divided into two parts in the ratio of eighty percent to twenty percent for training and testing datasets respectively. The training dataset was used to train an SVM classifier with the strategy of 10-fold cross validation which resulted in an SVM model. The Bayesian optimisation method was used to optimise the model by retraining and further selecting all possible values of the hyperparameters. This resulted in an optimised model. The SVM and the optimised SVM were tested with the testing dataset separately and further evaluated using the standard evaluation metrics.

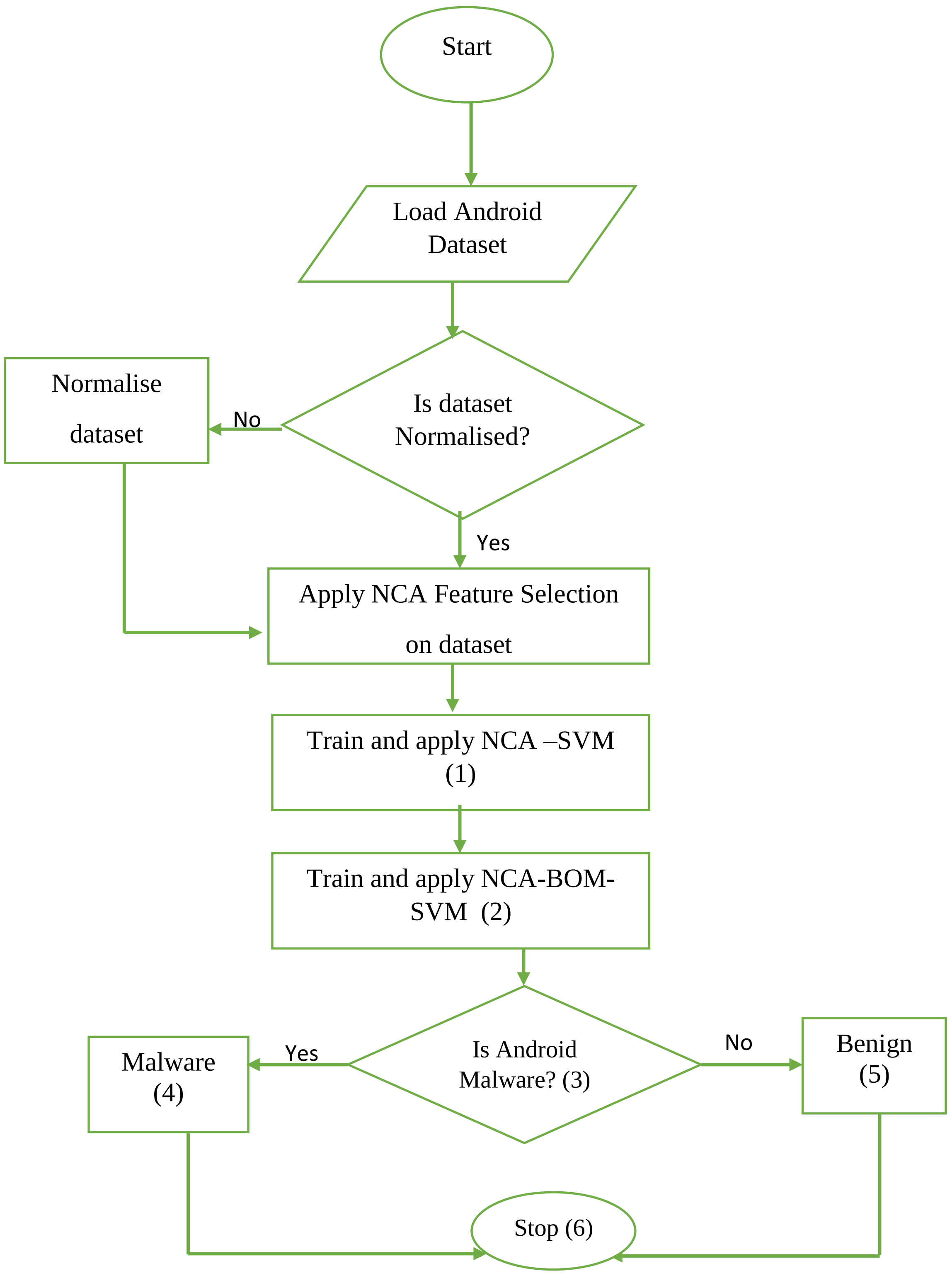

Figure 3.1 Detection Model

3.2 Data Collection

This research work adopted and deployed a standard and multi-dimensional android dataset from drebin which consist of thousands of data points arranged in rows and columns. The dataset has feature vectors of 215 attributes with 15,036 observations. The dataset was extracted from Drebin project with 5,560 malware apps and another 9,476 benign apps, all totalling 15,036 applications. Invariably, the dataset has 15,036 rows and 215 columns or features, in addition to the class label whose entries are either malware or benign. The dataset came with a supporting file that has the description of the feature vectors (attributes) which was extracted from the static code analysis of the Android apps. In order to have a balanced dataset to work with, the research work used the 5560 malware instances alongside the 5561 benign that was randomly selected from the 9476.

Table 3.1 Sample of features Category

onServiceConnected	API call signature
Transact	API call signature
bindService	API call signature
SEND_SMS	Manifest Permission
READ_PHONE_STATE	Manifest Permission

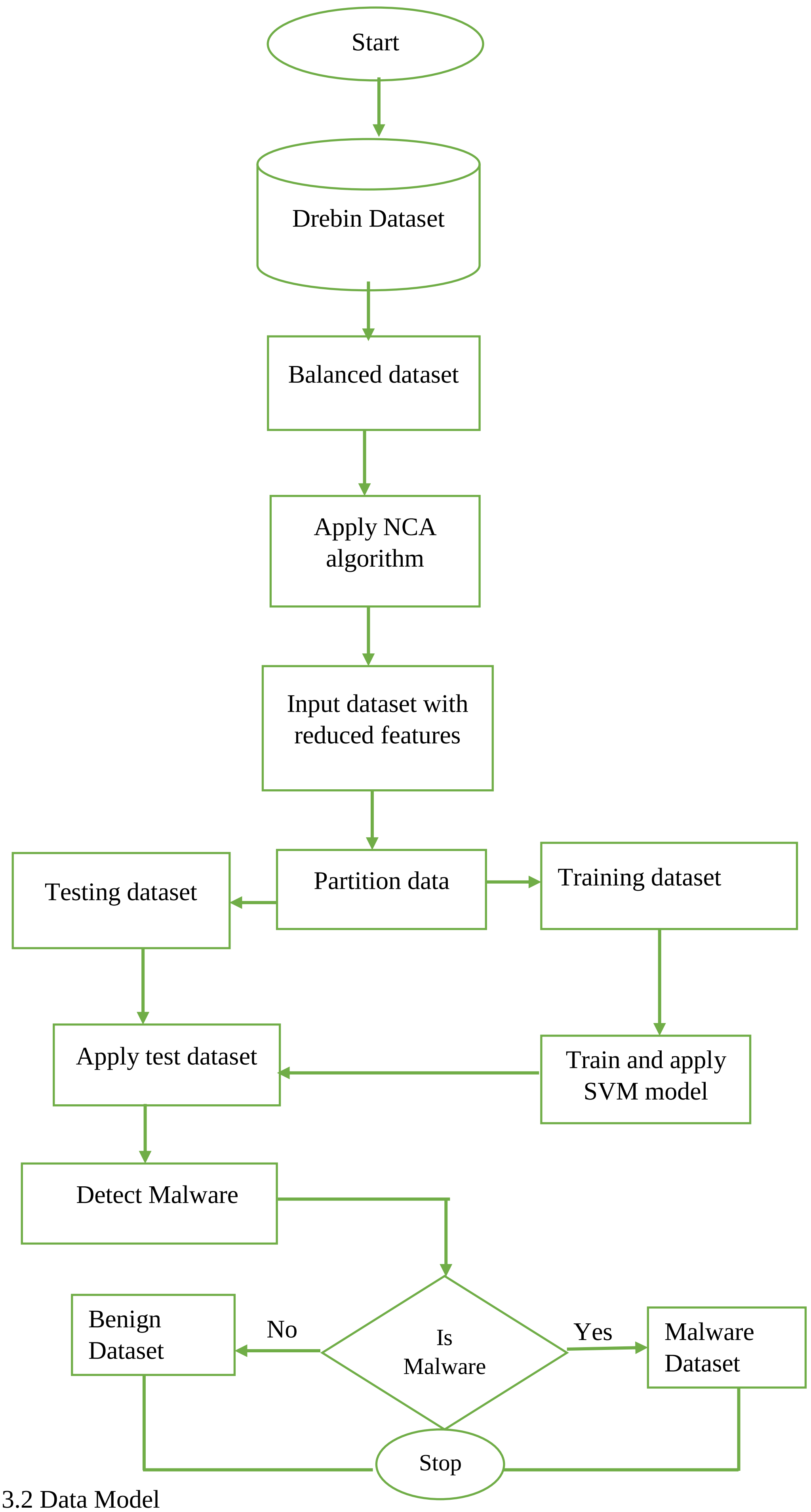

Figure 3.2 Data Model

3.3 Data Model

The data model as captured in figure 3.2 shows the flow of the data all through the research work. The original dataset was obtained bogus from drebin with large number of features and instances as explained in section 3.2. This drebin dataset had its curse of dimensionality reduced with the neighbourhood component analysis (NCA). It was reduced to 37 relevant and manageable features. This NCA dataset was divided into training and testing datasets in the ratio 80 to 20 respectively. NCA training dataset was used to the NCASVM model and the NCA-BOM-SVM model. While the NCA testing dataset was used to test the performance of two models. After this classification process, subsequent detection, the malware data formed another dataset and same for benign dataset.

3.4 Feature Selection

Feature selection is the technique deployed in machine learning for selecting a subset of input features that are most relevant to which is to be predicted. Apparently, most dataset come with less important, irrelevant, and redundant features which often distract and mislead machine learning algorithms, possibly resulting in lower or reduced prediction performance. Feature selection algorithms deploys techniques that reduces the curse of data dimensionality by searching for and selecting subset of predictor variables (measured features), that will optimally model the predicting class or the measured responses, subject to constraints like the size of the subset, or the excluded or required features. Invariably, it is quite desirable to expunge these irrelevant and redundant features before building models, and only utilise features that will result in best performing models. Applying feature selection algorithm comes with several benefits including removing irrelevant and less important features that have little or no contribution to building the model. It discards the non-trivial features thus reducing the complexity of the data. It is expedient to note that even when all features may contain information about the response variable and relevant, engaging in model building using all the features can denigrate the prediction performance.

These processes will start with the generation of several subsets from the original dataset, evaluations of each of the subsets based on the stopping criterion or criteria, and the validation of result, which will be the resulting optimised features. The subset generation covers the search by the feature selection algorithm for optimised features. The search direction could be backward or forward or bidirectional, depending on the algorithm deployed. The subset evaluation process entails defining fitness functions to determine the optimal features, and computing the fitness value. The best or appropriate fitness value, depending on the earlier set criterion will be chosen, having met the condition for stopping the subset generation process.

Sequential Feature Selection (SFS) algorithm, Neighbourhood Component Analysis (NCA) algorithm, Maximum relevance minimum redundant (MRMR) algorithm, and the relief algorithm were deployed on the dataset to carry out feature selection, in order to obtain the optimised features. Depending on the algorithms per time, the feature selection involved several processes and iteration and four separate optimised datasets were obtained from each of the algorithms.

3.4.1 Sequential feature selection algorithm implementation

The sequential feature selection is one of the feature selection algorithms that was deployed in this research work. On the Matlab platform where this algorithm was implemented, the function sequentialfs was used to achieve the purpose of the algorithm.

The function md = sequentialfs(fun,A B) selects a subset of the features from the predictor data matrix A, that best predict the class values or response class in B, by sequentially selecting features until there no improvement in the prediction. The A is a data matrix with 215 features and 10,121 observations, while B is a column vector which is the class variable, with 10,121 observations. The criterion value is defined by a function handle called the fun and it is used to select features and also determine when to halt the process. The features which are finally chosen are indicated or output by the logical vector called the md.

The algorithm begins with the sequentialfs creating a candidate or potential feature subsets by sequentially adding each feature that is not yet selected. A k-fold cross-validation is carried out on the selected subset by the sequentialfs which repeatedly calls the function handler fun with different training and testing subsets of A and B, obtained from the cross-validation. The criterion value which determine the chances of choosing a feature is obtained as –

Criterion = fun(ATrain, BTrain, Atest, Btest)

Criterion is a scaler value which the function handler fun returns each time it is called. Fun uses ATrain and BTrain to train and fit a model, then predicts the values of Atest using the model, and eventually returns some measure of distance or loss, of these predicted values from Btest. The criterion value for each selected feature set is obtained from the cross-validation by sequentialfs summing the values returned by fun and dividing it by the total number of test observations. From the mean criterion value of each selected feature subset, the sequentialfs picks the feature subset that minimises this value. This process is repeated until adding more features does not devrease the criterion value.

The flowchart of the feature selection obtained from the sequential feature selection algorithm is given in figure 3.3.

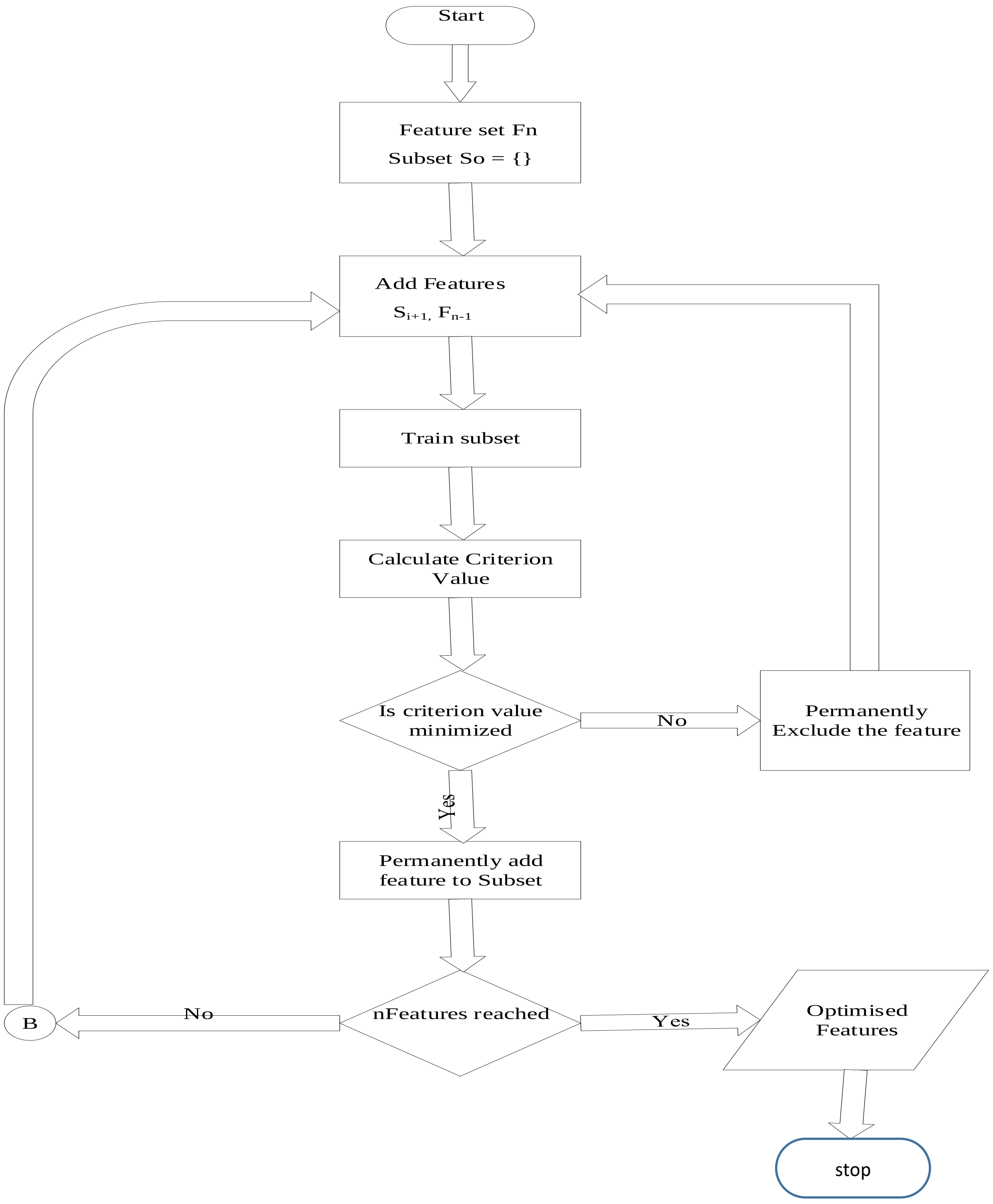

Figure 3.3 Feature Selection Flowchart

3.4.2 MRMR algorithm implementation

The MRMR was implemented with the Matlab in this research work. The fscmrmr is the function that was used to achieve its purpose. This function ranks all the features in the predicting dataset using the response class. The function [idx, scores] = fscmrmr(X,Y, Name, Value) has the predictor variable as X, the response variable as Y, Name specifies additional options as name-value pair arguments, and returns idx which contains the indices of predictors ordered by predictor importance, alongside the predictor scores. The importance of the predictor is determined by the score. More so, a drop in the feature importance score shows the confident of the feature selection. This entails that when value of the next important feature is much smaller than the score value of x, the algorithm is confident of selecting the preceding feature. The optimal set is obtained by selecting features in the order of importance from the score values of the various features.

Using fscmrmr function, MRMR algorithm can be implementation and rank features importantly through the following steps.

1) The feature with the largest relevance is selected and added to an empty set S.
2) Search for the features that have non-zero relevance and zero redundancy from the compliment of S, S^c.
    a. If there is no non-zero relevance and zero redundancy in S^c, jump to step 4
    b. Else pick the feature that has the largest relevance, and add to the set S.
3) Do step two again until the redundancy is not zero for all features in S^c.
4) Select the feature with the largest MIQ values with nonzero relevance and non-zero redundancy in S^c, and add the selected features to the set, S.
5) Do step 4 again until the relevance is zero for all features in S^c.

6) Add the features with zero relevance to S in random order.

3.5 Selection of the Training and Validation Data

The optimised dataset that was obtained from the feature selection process were treated separately since the essence of the feature selection is to test the strength and viability of the feature selection algorithms in generating a representative subset capable of enhancing model performance. Each of the datasets were divided to two part in the ratio of eighty percent to twenty percent for training dataset and test dataset respectively. The training dataset was used for the training of the classifiers with the k-fold cross-validation strategy. A 10-fold cross-validation was used which split the training dataset into 10 splits. Each split was further divided into 10. During this process, each data subset for each split had an opportunity of being a test/ validation set while the other nine was used was used for training. This entails that this particular process will be repeated until each of the groups has a chance to be used as the test set. Invariably, for ‘k’-fold cross-validation, the process will be repeated for ‘k’ unique times.

3.6 Training and Testing of the Classifier

The SVM classifier was trained and validated with the optimised datasets from the feature selection algorithms. The 10-fold cross-validation strategy was used in order avoid the challenge of overfitting. More so, the cross-validation gave the model the generalizing to unseen data before the testing phase. The model was tested with the test dataset and its performance thoroughly evaluated using the standard classification metrics. Considering the aim and objectives of this research, the focus of evaluation of the model’s performance was on its ability to seamlessly detect malware presence.

```
1: Start
2:  Initialize data (yTraining, xTraining, yTesting, xTesting)
3: Set partition as k = 10-fold cross-validation
4: Initialize SVM default parameters
        Set BoxConstraint = 1
        Set Standardize = true
        Set kernel = linear
5: Train SVM model with xTraining, yTraining
        For k= 1to 10
        Train & validate SVM
        Return SVMModel
6: Test SVMModel with yTesting, xTesting
      Return metric results
7: Compute & evaluate metrics results
8: Visualise metric results
9: Stop
```

Figure 3.4 SVM Classifier Pseudocode for Android malware Detection

3.7 Optimisation of the Model

This research work deployed Bayesian optimization method for optimization on the resulting SVM model, where several internal parameters and hyperparameters were combinatorically tried and modelled. These variations provided the necessary tunings to the optimised SVM. The following parameter are the optimizable hyperparameters that were tuned

- ❖ Kernel – values varies from Gaussian, Linear. Quadratic and Cubic
- ❖ Box Constraint Level - values varies between [0.001 and 1000]
- ❖ Kernel Scale -values varies between [0.001 and 1000]

- Multiclass method – Either one-vs-one or one-vs-all
- Standardise data – Either true or false

Other important optimisation options that were used include-

- Acquisition function
- Training time limit
- Iterations
- Maximum training time I seconds

```
1: Start
2: Initialize data (xTraining, yTraining, xTesting, yTesting)
3: Set partition as k=10 fold cross-validation
4: Initialize & pre-set some SVM parameters & hyperparameters
5:          Set Acquisition function = expected-improvement plus
6:          Set Optimize Hyperparameters = auto
7:          Set Standardize = true
8:          Set kernel = gaussian
9:          While best feasible point is not obtained
10:              Train & validate BOMSVM model
11:              Get feasible points
12:                  Get BoxConstraint values
13:                  Get KernelScale values
14:                  Get objective function value
15:                  Get estimated objective function values
16:                  Get function evaluation time
17:              If estimated best feasible point is reached
18:                  Get optimal BoxConstraint value
19:                  Get optimal KernelScale values
20:              Else continue training & optimization
22:          End while
```

```
23: Return pre-set parameters & hyperparameters, optimal BoxConstraint value, optimal…
24: KernelScale value, optimal objective function value, estimated objective…
25: value, function evaluation time
26: Test BOM-SVM model with xTesting, yTesting
27: Return metric results
28: Compute & evaluate metrics results
29: Visualize metric results
30: Stop
```

Figure 3.5 BOM-SVM Classifier Pseudocode for Android Malware detection

3.8 Optimised NCA-BOM-SVM Model

The optimization of SVM with Bayesian optimization method is the NCA-BOM-SVM. The sequential implementation of this model is followed from the flow chart in figure 3.1 and the pseudocode in figure 3.5. The android dataset with 215 features was trained on the neighbourhood component analysis for the feature selection process. This process resulted in a dataset (NCA Dataset) with 37 highly relevant features. The NCA dataset is divided into the training and testing data in the ratio of 80 to 20.

The essence of optimising SVM is to minimize the cross-validation loss and optimize some hyperparameters at the best feasible points that will develop a model with optimal performance. However, some parameters and hyperparameters of SVM were pre-set to some value. The kernel was pre-set to gaussian since the Bayesian optimization method makes use of the gaussian function. Apart from the pre-set parameters and hyperparameters, other hyperparameters referred to as the optimizable hyperparameters were set to auto. This include the box constraint and the kernel scale. With these selected values of these parameters and hyperparameters, an objective function was created and was varied all through the experiment until the best feasible points were obtained for auto hyperparameters. The SVM was trained using the NCA training dataset with recourse to the hyperparameters adopting the 10-fold cross-validation strategy. The acquisition function which is one of the parameters determines the values and points of the auto hyperparameters that have not been modelled and further picks the next possible values to be modelled. The training and optimisation of SVM came to an end when the best feasible points for the auto-set hyperparameters were obtained These points are captured in table 4.4. This is the optimised SVM model called the NCA-BOM-SVM model. The performance of the optimised model is further tested with the NCA testing dataset and the results were obtained and evaluated as captured on table 4.6 while the minimum classification error graph for the optimisation is figure 4.5. The evaluation of the NCA-BOM-SVM performance was done based the standard evaluation metrics and the results are capture on table 4.6.

3.9 Evaluation of Classification Metrices

Performance evaluation of a classification while developing a machine learning model is undeniably a key step and essential part that cannot be evaded. As such, measuring the performance of a trained model, is absolutely very important. Both the adaptive and non-adaptive capacity of a machine learning model is determined by the ability of the model to perfectly generalise on the unseen data.

Performance evaluation metrics is the tool deployed or medium through which improvement could be done on the overall predictive power of a model. This metrics, which provides information concerning model performance, is used to monitor and further measure the performance of the model. The proper evaluation of the ML model, builds in confident and strength in the model's generalisation accuracy in subsequent predictive capacity. It evaluates the model's performance which tells how good or bad the classification is. Each of the evaluation metric evaluates the model in different ways.

3.9.1 Confusion matrix

This is a matrix representation, with tabular visualisation of the real classification labels and the model predictions, being the prediction result of classification that is often used to describe the performance of the model on a stipulated set of test data at a known time. This is a simple matrix to decipher, whose row represents the instances in an actual class.

Actual	Predicted	
	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

Figure 3.6 Confusion Matrix for the proposed model

Each prediction in the confusion matrix represents an evaluation factor and can be one of the following outcomes-

- ❖ *True positive (TP)* – Predicted and actual values are positive.
- ❖ *True negative (TN)* - Predicted and actual values are negative.
- ❖ *False positive (FP)* – Predicted value is positive whereas actual value is negative. This is equivalent to type I error.
- ❖ *False negative (FN)* – Predicted value is negative whereas actual value is positive. This is equivalent to type II error.

3.9.2 Accuracy

This is the ratio of the number of the correct or right predictions to total predictions. So, it tells how the classifier often make correct prediction. Accuracy is put to use mainly and works well when the classes are equal in size, that is, number of samples belonging to each class are equal.

Accuracy = (TP+TN)/total number of predictions

Misclassification Rate (Error Rate) – This tells how often the classifier predicts incorrectly or misclassify. This is a measure of the failure rate in terms of classification by the classifier.

Misclassification Rate = (FP+FN)/total of predictions made

3.9.3 Precision

Precision stipulates the ratio of right or correct predictions to overall actual positive prediction. It tells how often is it correct, when the classifier predicts true or yes.

Precision=TP/predicted yes; Precision = TP/TP+FP

Precision is often put to use when there is a class imbalance, thus, accuracy becomes unreliable metric for measuring the model performance.

3.9.4 Recall or sensitivity

Recall offers the measurement metrics that stipulates how often a model predict positive or yes, when the actual value is positive. This metric is a measure of the true positive rate (TPR).

Recall (TPR) = TP/Actual positive

It is best used when the sample dataset is imbalance.

3.9.5 F1_score

This is the harmonic mean of the precision and recall.

F1_Score = 2 x (Precision x Recall)/(Precision + Recall)

3.9.6 Specificity

Specificity is the true negative rate (TNR) or the proportion of true negatives to everything that should have been classified as negative.

Specificity (TNR) = TN/Actual negative

3.9.7 Receiver operating characteristics (ROC) curve

ROC measures the area under the ROC curve. This measurement is done by plotting the true positive rate against the false positive rate. This plot produces the ROC curve, which allows the model designer to visualise the trade-off between the true positive rate and the false positive rate.

CHAPTER FOUR

4.0 RESULTS AND DISCUSSIONS

4.1 Feature Selection Results

4.1.1 Feature selection -MRMR

The Table 4.1 displays the result of feature selection by MRMR algorithm ranking the predictors and their corresponding weights.

Table 4.1 Feature ranking based on weight by MRMR algorithm.

Predictors ranking	Weight of Predictor	Predictors ranking	Weight of Predictor
1	0.2099	60	0.0028
127	0.2011	63	0.0028
169	0.1247	15	0.0027
99	0.0192	159	0.0027
80	0.0190	73	0.0025
42	0.0098	23	0.0025
86	0.0096	61	0.0023
109	0.0078	46	0.0022
14	0.0075	56	0.0020
75	0.0056	54	0.0020
27	0.0042	24	0.0019
106	0.0036	9	0.0019
13	0.0035	31	0.0019
51	0.0032	7	0.0019
126	0.0029	66	0.0018

The Figure 4.1 is the graphical representation of the predictors ranking and their corresponding weights obtained from the MRMR algorithm.

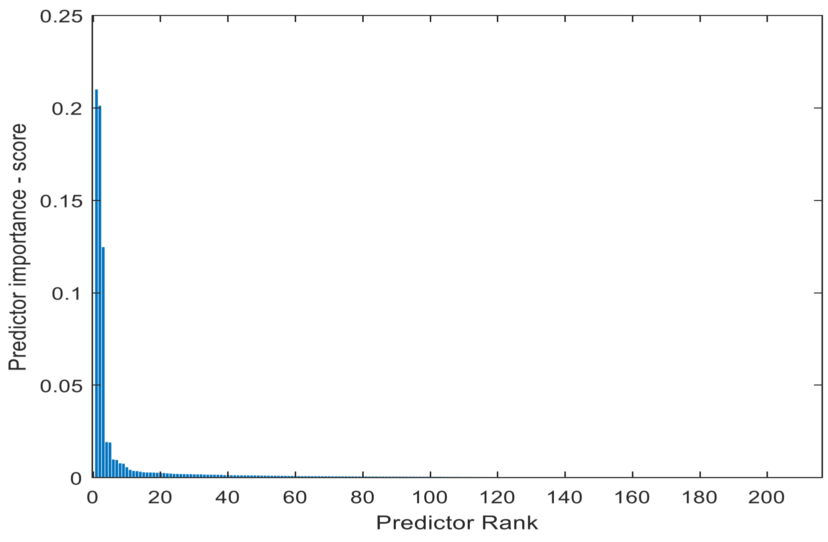

Figure 4.1 Feature ranking by MRMR

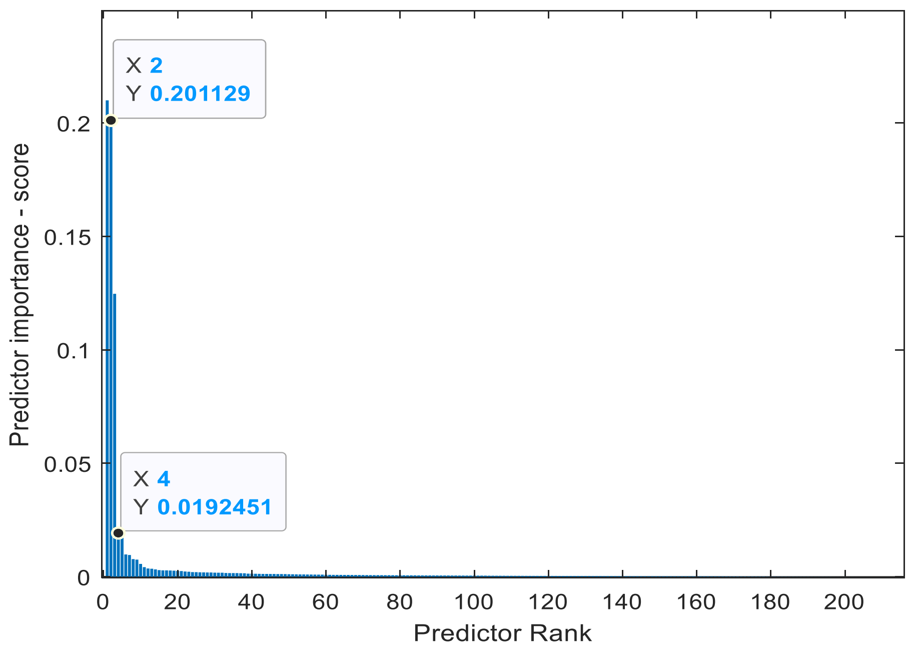

Figure 4.2 Feature ranking by MRMR

4.1.2 Feature selection - NCA

The Table 4.2 is the result of the selected predictors according to their columns and their corresponding feature weights using the neighbourhood component analysis.

Table 4.2 Feature ranking based on weight by NCA

Predictors	Feature Weights	Predictors	Feature Weights
X7	2.70384	X23	1.31786
X16	2.55435	X26	1.35393
X29	1.69903	X22	1.4987
X35	1.97921	X33	1.24626
X43	1.79244	X42	1.10225
X47	1.59551	X48	1.20294
X95	1.70993	X51	1.14678
X102	1.79552	X62	1.1376
X122	1.92739	X64	1.45422
X138	1.151621	X69	1.30107
X172	1.8032	X76	1.37174
X181	1.60786	X85	1.12156
X207	1.66404	X93	1,29766
X211	1.65918	X97	1.20723
X9	1.01797	X100	1.228
X8	1.09952	X123	1.28589
X13	1.10374	X197	1.20545
X15	1.19292	X203	1.03951
		X212	1.20908

The Figure 4.3 gives the graphical representation of the predictors ranking and their corresponding feature weights by the neighbourhood component analysis algorithm.

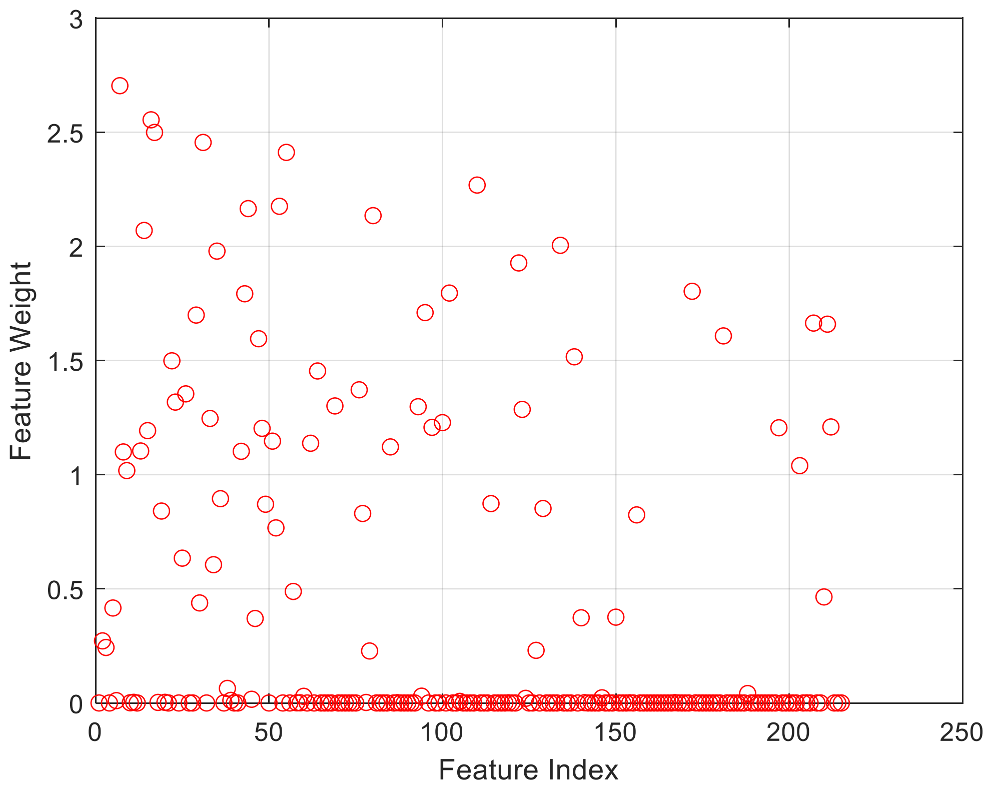

Figure 4.3 Feature ranking by NCA algorithm

4.1.3 Feature Selection -Sequential Feature Selection (SFS)

The Table 4.3 shows the result of the Sequential Feature Selection algorithm where some predictors were chosen based on their criterion value.

Table 4.3 Chosen features by SFS based on criterion value

Chosen Features	Criterion Value	Chosen Features	Criterion Value
1	0.000206714	106	5.81947e-05
14	0.000134753	86	5.75111e-05
7	0.000113757	27	5.71202e-05
8	9.61817e-05	62	5.68277e-05
69	8.94436e-05	61	5.64369e-05
60	8.3585e-05	135	5.62415e-05
56	7.92889e-05	24	5.61438e-05
42	7.58702e-0.5	43	5.3994e-05
94	7.22563e-05	155	5.2822e-05
202	6.80572e-05	80	5.25308e-05
28	6.62992e-0.05	11	4.93086e-05
67	6.52257e-05	5	4.67681e-05
37	6.41518e-05	70	4.52059e-05
66	6.31748e-05	172	4.38396e-05
82	6.22962e-05	53	4.30591e-05
119	6.13201e-05	49	3.97366e-05
139	5.98552e-05	192	3.90537e-05
145	5.8976e-05	194	3.87615e-05

4.2 Results of Optimised NCA-BOM-SVM

Table 4. 4 Optimised SVM hyperparameters that were tuned and values

Kernel function	**Box constraint level**	**Kernel Scale**	**Acquisition Function**	**Standardize Data**	**Optimiser**
Gaussian	68.1216	Auto	Expected improvement per second plus	true	Bayesian Opt
Gaussian	1.7529	Auto	Expected improvement per second plus	true	Bayesian Opt
Gaussian	0.5212	Auto	Expected improvement per second plus	false	Bayesian Opt

Pertinent to mention that the correct combination of the various hyperparameters alongside the modelling of the different values of the hyperparameters led to the optimisation of the SVM classifier. The Bayesian Optimization method made it possible for the different values to be modelled.

Table 4.5 Default SVM Parameters and values

Kernel	Box Constraint	Kernel Scale	Standardize data
Linear, rbc	1	0	False

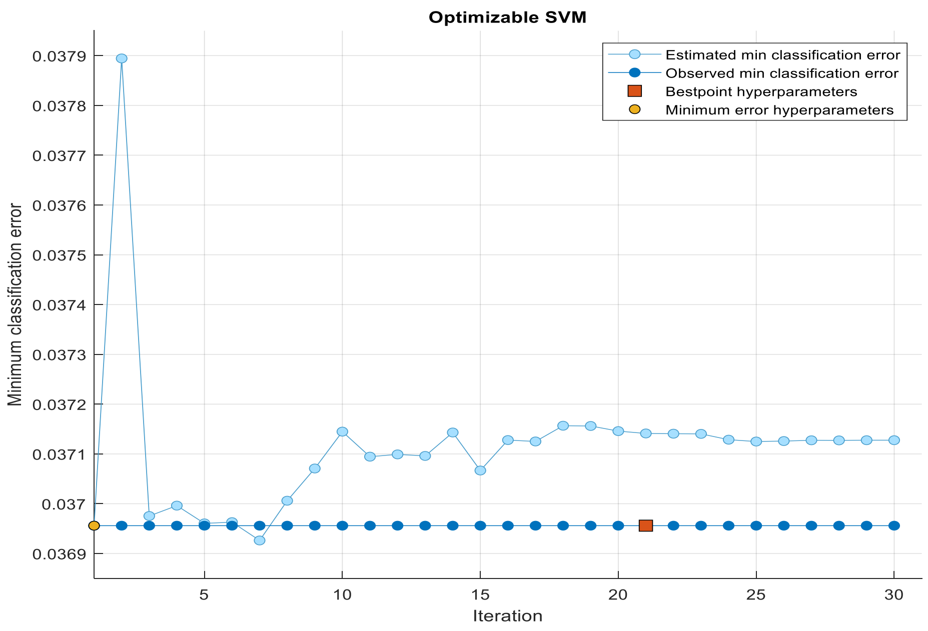

Figure 4.4 Minimum classification error graph for optimised SFS-BOM-SVM

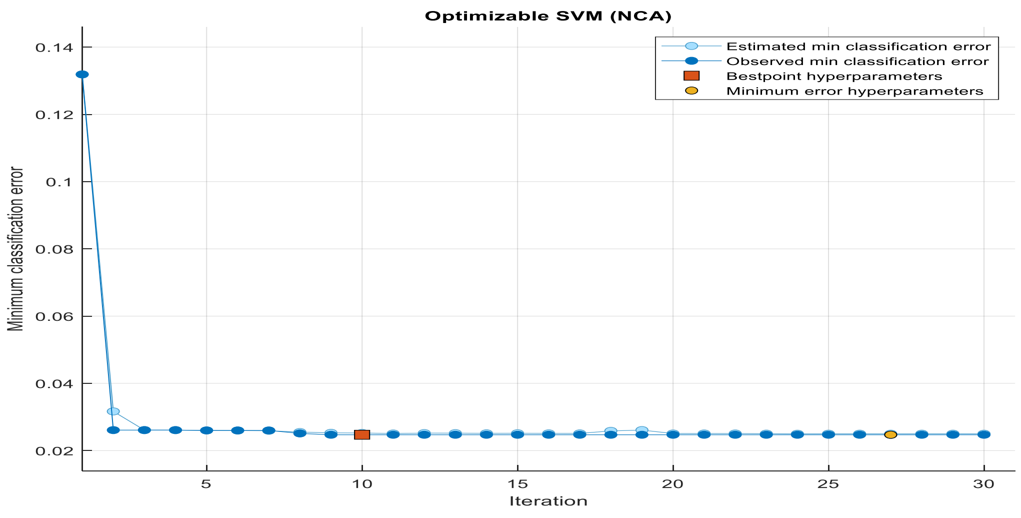

Figure 4.5 Minimum classification error graph for optimised NCA-BOM-SVM

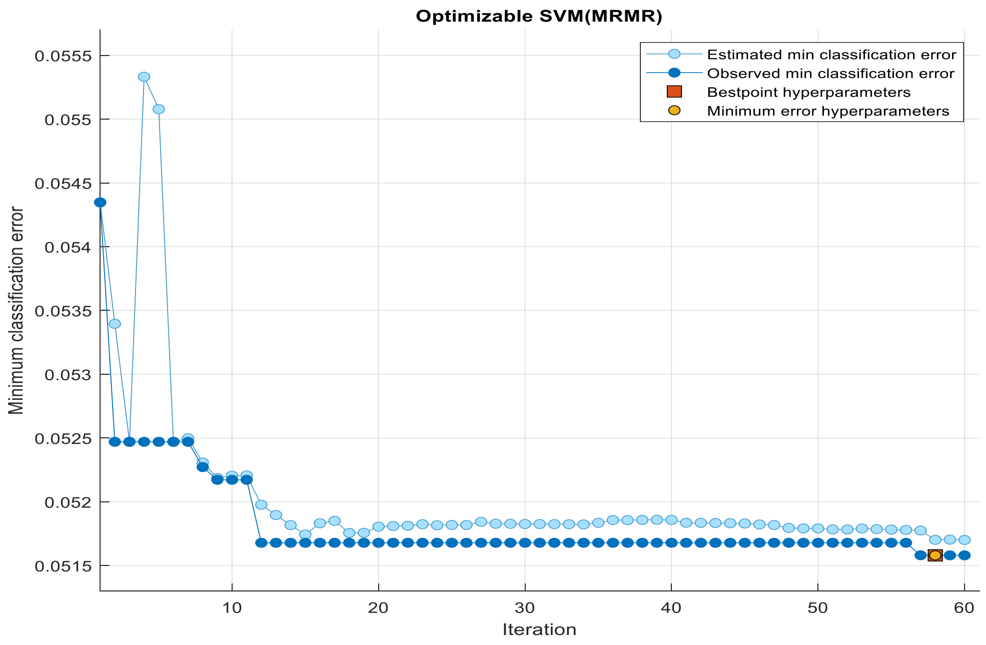

Figure 4.6 Minimum classification error graph for optimised MRMR-BOM-SVM

Table 4.6 Performance Analysis of SVM and Optimised SVM (BOM-SVM)

Algorithm	Accuracy %	False Alarm Rate	Precision	Error Rate	Recall	F1_Score
SFS-SVM-Linear	95.1	0.05	94,5	0.048	96.2	95.34
SFS-BOM-SVM	96.3	0.036	96.4	0.036	96.6	96.49
NCA-SVM Quadratic	95.8	0.037	96.3	0.04	95.6	95.9
NCA-BOM-SVM	97.8	0.021	97.9	0.02	97.9	97.9
MRMR-SVM- - Quadratic	94.4	0.067	93.3	0.055	96.2	94.7
MRMRBOM-SVM	94.9	0.065	93.5	0.05	96.9	95.2

Table 4.7 Comparison of Proposed Optimized SVM Classifier with Baseline Literatures

Reference	Classifier/Method	Accuracy %	False Alarm	Precision	Recall	F1_Score
Proposed Optimised SVM	NCA-Optimised SVM	97.8	0.021	97.9	97.9	97.9
Rana et. al. (2018)	Random Forest	94.33	Not reported (NR)	94	95	94
	SVM	90.74		91	91	91
	DT	91.78		88	94	91
	ERT	93.66		93	93	93
	BAGG	93.71		94	94	94
Rana et. al. (2018)	DT	96.13	Not reported (NR)	96	96	96
	RF	97.24		98	97	97
	Gradient Boosting	93.68		94	93	94
	Ext Randomised	96.97		97	97	97

Li et. al. (2018)	DroidDeepLearner / Deep Learning	90	(NR)	(NR)	(NR)	(NR)

Further, the table 4.7 is a comparative analysis between the proposed solution and solutions from the base line literatures with reference to accuracy, precision, recall, fi_score and possibly the false alarm rate as the standard evaluation metrics. From Table 4.9, the proposed solution displays mastery and strength over the methods in the base line literatures considering the accuracy, recall, precision and f1_score.

4.3 Results Discussion

Table 4.1 displays the thirty features with their corresponding weights, that were selected for the experiment as ranked by the MRMR algorithm from the 215 features from the original dataset based on their order of importance. Similarly, Table 4.2 and table 4.3 show the selected features with corresponding weights and criterion values by the NCA and SFS algorithms respectively, from the 215 features from the original datasets. The weights in Table 4.2 show the importance of the features whereas the criterion value is the selection criteria for the feature to be added to the subset solution. For every selected feature in Table 4.3, its criterion must have been minimised else would have been dropped by the SFS algorithm. Figures 4.3 and 4.4 show the graphical representations of the features based on their ranking by the MRMR algorithm. Similarly, Figure 4.5 shows the scattered plots of the features based on their weights by NCA algorithm. The plots points with zero weights show how less relevant the features are.

The results show that there is considerable benefit for deploying feature selection on datasets for classification purpose. The various feature selection algorithms used their methods to rank and select features they felt were representative of the original dataset. The algorithms were able to reduce the features from 215 to the numbers mentioned above. Only the SFS algorithm selected a certain number of features which was 36, while others only ranked the features and gave the researcher the discretion of the number of features to be chosen.

Table 4.4 shows the optimal points that the Bayesian optimization method chose as the points that optimised the SVM model which gave better performance. These optimal points were chosen after several combinations and modelling of all the possible values of the hyperparameters. Figures 4.6, 4.7 and 4.8 shows the minimum estimation graphs for the MRMR, NCA and SFS optimised dataset. The red points on the graphs represents the minimum error rates for the optimised SVM model. Table 4.6 shows the performance analysis of the SVM and optimised SVM, with consideration to the dataset from the feature selection algorithms.

It was observed that the datasets from the feature selections performed above acceptable level but the NCA outperformed all others as it produced alongside the optimised SVM a higher accuracy rate of 97.8%, precision of 97.9% and recall of 97.9, with lower error rate of 0.02 and false alarm rate of 0.021 compared to the combination of other feature selection algorithm with the optimised SVM as shown in table 4.8. NCA-SVM optimised model gives the proposed solution which is better that the result obtained from the SVM model without feature selection.

Table 4.7 shows the comparative analysis of the proposed solution with some baseline literature that used the same original dataset for their experiment. The results as displayed from the table shows the proposed solution outperforming the results from the baseline literatures. The proposed solution has an accuracy of 97.8%, precision of 97.9%, recall 97.9% and f1_score of 97.9%, compared with the best result from the baseline literature from Rana et. al. (2018) that obtained accuracy of 97.24%, precision of 98%, recall of 97% and f1_score of 97%.

CHAPTER FIVE

5.0 CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

There is no doubt that there is an exponential increase in the reported cases of Android malware. More so, worthy of concern is the level of sophistication in malware development which adopts different means including the code obfuscation. However, this research work was proposed to stem this menace by developing an ML model that will detect Android malware on mobile Android platforms. The research work deployed different feature selection algorithms which were the sequential feature selection algorithm, relief algorithm, neighbourhood component algorithm and the minimum redundant maximum relevant algorithm for feature selection process. These algorithms used their different methods and operations to rank, provide weights and criterion values to the features, which were considered for evaluation and selection and eventually used for training, validation and testing the SVM and optimised SVM. The process of feature selection indeed helped to reduce the curse of dimensionality of the dataset while still keeping its representation properties. This process saw the number of features reduced in each case from 215 in the original dataset.

During the training and validation, the strategy of k-fold cross-validation helped to reduce overfitting and to enhance the generalization ability of the model to predicting unseen data. This is because there was testing in each split of the cross-validation process. SVM has some default values that are used while developing the model. But the Bayesian optimization method offered the opportunity to model almost every point in the optimisable hyperparameters and also provided the optimal values for the hyperparameters where the best solution was obtained. However, based on the displayed results obtained from the evaluation of the performance of the model, there is no doubt that the proposed solution has demonstrated strength in the detection of android malware.

5.2 Recommendations

The following recommendations are made in furtherance any future from the spinoff of this research work.

1. SVM is a promising ML classifier that can be enhanced for super accuracy for malware detection and similar challenges
2. Possibilities of exploring a means of having an ensemble of two or more feature selection algorithms for more optimal output.

5.3 Contribution to Knowledge

This research work made the following contributions to knowledge-

1. Development of a tool for detection of Android malware using Bayesian optimised SVM with neighbourhood component analysis algorithm.

REFERENCES

Adebayo, O. S., & Abdul Aziz, N. (2019). Improved Malware Detection Model with Apriori Association Rule and Particle Swarm Optimization. *Security and Communication Networks*, *2019*, 1–13. https://doi.org/10.1155/2019/2850932

Adebayo, O. S., & Aziz, N. A. (2014). Techniques for analysing Android malware. *The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*. Published. https://doi.org/10.1109/ict4m.2014.7020656

Adebayo, O. S., & AbdulAziz, N. (2014). Android malware classification using static code analysis and Apriori algorithm improved with particle swarm optimization. *2014 4th World Congress on Information and Communication Technologies (WICT 2014)*. https://doi.org/10.1109/wict.2014.707731

Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2020). DL-Droid: Deep learning based android malware detection using real devices. *Computers & Security*, *89*, 101663. https://doi.org/10.1016/j.cose.2019.101663

Android malware detection technique via feature analysis, Journal of Engineering Android, 2016 INTECH Innovation Conference

Aneja, L., Babbar, S. (2018) Research trends in malware detection on Android devices.

Bhatia, T., Kaushal, R. (2016). Malware Detection in Android based on Dynamic Analysis,Canfora, G. "Effectiveness of opcode ngrams for detection of multifamily android

Chavan, N., di Troia, F., & Stamp, M. (2019). A Comparative Analysis of Android Malware. *Proceedings of the 5th International Conference on Information Systems Security and Privacy*. Published. https://doi.org/10.5220/0007701506640673

Coronado-De-Alba, L. D., Rodríguez-Moto, A., Escamilla- Ambrosio, P. J. (2016). Feature

Cortes, C. & Vapnik, V. Support Vector Networks, Machine Learning, vol. 20, no. 3, pp.273-297, 1995

Dubey, A., Misra, I. (2013). Android security attacks and defenses, Parkway NW: CRC Press Taylor & Francis

Elenkov, N. (2015). Android Security Internals, An In-Depth Guide to Android's Security Architecture, San Francesco: William Pollock

Engineering Science and Technology, 14(3), 1572 – 1586

Gibert, D., Mateu, C., Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, Journal of Network and Computer Applications, (153), doi: 10.1016/j.jnca.2019.102526

Jami, Q., Shah, M.(2016). Analysis of Machine Learning Solutions to Detect Malware in

K. Xu, Y. Li, R. H. Deng, and Deng, "Iccdetector: icc-based malware detection on android," IEEE Transactions on Information Forensics and Security, vol. 11, no. 6, pp. 1252–1264, 2016.

Khan, N., Abdullah, J., & Khan, A. S. (2017). Defending Malicious Script Attacks Using Machine Learning Classifiers. *Wireless Communications and Mobile Computing*, *2017*, 1–9. https://doi.org/10.1155/2017/5360472

malware," in Proceedings of the 2015 10th International Conference on Availability, Reliability and Security, IEEE, Toulouse, France, August 2015.

Martin, I., Hermader, J. A., Munoz A. (2018). Android characteristic using metadata machine doi: 10,1155/2018/5749481

Memon, L. U., Bawany, N. Z., Shamsi, J. A. (2019). A comparison of machine learning

Narayanan, A. (2018) "APK2VEC: semi-supervised multi-view representation learning for Profiling Android applications," in Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), IEEE, Beijing

Ng, A. P.. Chiew, K. L., Ibrahim, D. H. A., Tiong, W. K., Sze, S. N., Musa, N.(2018)

Rana, M. S., Gudla, C., & Sung, A. H. (2018). Evaluating Machine Learning Models for Android Malware Detection. *Proceedings of the 2018 VII International Conference on Network, Communication and Computing - ICNCC 2018*. Published. https://doi.org/10.1145/3301326.3301390

Rana, M. S., Rahman, S. S. M. M., & Sung, A. H. (2018). Evaluation of Tree Based Machine Learning Classifiers for Android Malware Detection. *Computational Collective Intelligence*, 377–385. https://doi.org/10.1007/978-3-319-98446-9_35

Ranveer, S., Hiray, S., (2015). SVM Based Effective Malware Detection System. International Journal of Computer Science and Information Technologies, Vol. 6 (4) , 2015, 3361-3365

Rathore, H., Agarwal, S., Sahay, S. K., Sewak, M. (2019). Malware detection using machine learning and deep learning Science and Technology, (2018) 78 – 90 Selection and Ensemble of Classifiers for Android Malware Detection, Proceedings of 2016 IEEE International Conference Springer Nature, (799), 629–642, doi:10.1007/978-981-10-8527-7_53.

Wang, J., Jing, Q., & Gao, J. (2019). SEdroid: A robust Android malware detection using selective ensemble, Proceedings of the 5th International Conference on Information Systems Security and Privacy.

Wang, Z., Cai, J., Cheng, S., & Li, W. (2016). DroidDeepLearner: Identifying Android malware using deep learning. *2016 IEEE 37th Sarnoff Symposium*. Published. https://doi.org/10.1109/sarnof.2016.7846747

Wen, L., Yu, H. (2017). An Android malware detection system based on machine learning. AIP Conference Proceedings, (1864), doi:10.1063/1.4992953

Wu, Q., Zhu, X., Liu, B(2021). A Survey of Android Malware Static Detection Technology Based on Machine Learning, Mobile Information Systems, ID 8896013, doi.org/10.1155/2021/8896013

Yang, M., Chen, X., Zhang, H., (2020). An Android malware detection model based on DT-SVM. Security and Communication Netwoks, (2020), doi: 10.1155/2020/8841233

Yuan, Z., Lu, Y., & Xue, Y. (2016). Droiddetector: android malware characterization and detection using deep learning. *Tsinghua Science and Technology*, *21*(1), 114–123. https://doi.org/10.1109/tst.2016.7399288