

# **SEDMDroid: An enhanced stacking ensemble framework for Android malware detection**

## **A PROJECT REPORT**

*Submitted by*

**ARYA PATEL [Reg No: RA1911031010084]  
ASISH KONDRAGUNTA [Reg No: RA1911031010125]**

*Under the Guidance of*

**Dr.K.A.VARUN KUMAR**

(Associate Professor, Networking and Communications)

*in partial fulfillment of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY  
in  
COMPUTER SCIENCE AND ENGINEERING  
with specialization in CYBERSECURITY**



**DEPARTMENT OF NETWORKING AND COMMUNICATIONS  
COLLEGE OF ENGINEERING AND TECHNOLOGY  
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY  
KATTANKULATHUR - 603 203**

**MAY 2023**



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY  
KATTANKULATHUR – 603 203**

**BONAFIDE CERTIFICATE**

Certified that this B.Tech project report titled **“SEDMDroid: An enhanced stacking ensemble framework for Android malware detection”** is the bonafide work of **Mr ARYA PATEL** and **Mr. ASISH KONDRAGUNTA** who carried out the project work under our supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

**DR K A Varun Kumar**

**SUPERVISOR**

Associate Professor

Department of Networking and  
Communications

**DR ANNAPURANI K**

**HEAD OF THE DEPARTMENT**

Professor

Department of Networking and  
Communications

**Signature of the Internal Examiner**

**Signature of the External Examiner**



Department of Networking and Communications  
**SRM Institute of Science & Technology**  
**Own Work\* Declaration Form**

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

**Degree/ Course : B.Tech - Computer Science and Engineering**

**Student Name(s) : Arya Patel  
Asish Kondragunta**

**Registration Number(s) : RA1911031010084  
RA1911031010125**

**Title of Work : SEDMDroid: AN ENHANCED STACKING  
ENSEMBLE FRAMEWORK FOR ANDROID  
MALWARE DETECTION**

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism\*\*, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that I / We have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

**DECLARATION:**

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

ARYA PATEL  
RA1911031010084

ASISH KONDRAGUNTA  
RA1911030031010125

## ACKNOWLEDGEMENT

We express our humble gratitude to **Dr C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr T.V.Gopal**, for his valuable support.

We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr Annapurani K**, Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our program coordinator **Dr A Suresh**, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for their inputs during the project reviews and support. We register our immeasurable thanks to our Faculty Advisor, **Dr P Supraja**, Assistant Professor, Networking and Communications Department, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr K A Varun Kumar**, Associate Professor, Networking and Communications SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship. She provided us with the freedom and support to explore the research topics of my interest. Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Networking and Communications Department staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support, and encouragement.

**Arya Patel [Reg No: RA191103010084]**

**Asish Kondragunta [Reg No: RA1911031010125]**

## ABSTRACT

The prevalence of the android operating system in mobile devices and other Internet-connected objects has led to a surge in malware attacks targeted towards it. To combat this growing issue, We present a framework for similar malware detection called SEDMDroid. Malware poses a significant threat to device security and the sensitive data stored within them, such as personal information. Our study aims to explore the prediction of malware in Android. The initial phases of our research involve data pre-processing and model selection, which includes to split the information into a train set as well as a test set with an ratio of “80” and “20”, respectively. In the third phase, we implement Principle Component Analysis for feature reduction and utilize prominent prediction models such as Random Forest and XGBoost algorithms to assess their impact on model accuracy. Finally, we analyze the results of the predicted outcome in the form of an AUC/RUC curve to evaluate our findings on the test set.

Other than that, the use of deep learning algorithms such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) can be explored to further improve the accuracy of android malware detection. These algorithms can be trained on large-scale datasets and can learn complex patterns and relationships between features, leading to more accurate predictions. Another area of improvement is the use of dynamic analysis techniques, such as behavioral analysis, to identify malware based on its actions rather than just its static characteristics. This can enhance the detection capabilities of android malware detection systems by identifying previously unknown malware that may not have been detected using static analysis techniques. Moreover, the integration of blockchain technology can provide an additional layer of security and transparency to the detection process, making it more robust and resistant to tampering. Overall, these advancements can significantly improve the effectiveness of android malware detection systems, thereby ensuring the security and privacy of user data.

# TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	TABLE OF CONTENTS	vi
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	x
1.	INTRODUCTION	1
	1.1 OVERVIEW	1
	1.2 PROJECT OBJECTIVE	2
	1.3 PROBLEM STATEMENT	2
	1.4 MOTIVATION	2
	1.5 PURPOSE	3
2.	LITERATURE REVIEW	4
	2.1 EXISTING SYSTEM	4
	2.1.1 DISADVANTAGES OF THE EXISTING SYSTEM	4
	2.2 PROPOSED SYSTEM	4
	2.2.1 ADVANTAGES OF THE PROPOSED SYSTEM	5
	2.3 LITERATURE SURVEY	5
	2.3.1 COMPREHENSIVE REVIEW OF RECENT LITERATURE FOR DETECTING MALWARE ON ANDROID DEVICES	5

3	<b>SYSTEM ARCHITECTURE AND DESIGN OF SEDMDroid</b>	11
3.1	ARCHITECTURE DIAGRAM	11
3.2	ER DIAGRAM	12
3.3	SEQUENCE DIAGRAM	13
3.4	ACTIVITY DIAGRAM	14
4	<b>IMPLEMENTATION OF SEDMDroid</b>	17
4.1	MODULES	17
4.2	MODULE DESCRIPTION	17
4.2.1	DATA SELECTION AND LOADING	17
4.2.2	DATA PREPROCESSING	17
4.2.3	SPLITTING DATASET INTO TRAIN & TEST DATA	18
4.2.4	REDUCTION FEATURE	18
4.3	CLASSIFICATION OF THE DATASET	18
4.3.1	RANDOM FOREST	18
4.3.2	XGBOOST	18
4.4	PREDICTION	19
4.5	OUTCOME GENERATION	19
5	<b>SYSTEM REQUIREMENTS OF SEDMDroid</b>	20
5.1	HARDWARE REQUIREMNT	20
5.2	SOTWARE REQUIREMENT	20

5.3	SOFTWARE DESCRIPTION	20
5.3.1	PYTHON	20
5.4	FEASIBILITY STUDY	22
5.4.1	ECONOMIC FEASIBILITY	23
5.4.2	TECHNICAL FEASIBILITY	23
5.4.3	BEHAVIORAL FEASIBILITY	23
5.5	TESTING OF PRODUCTS	23
5.5.1	UNIT TESTING	23
5.5.2	INTEGRATION TESTING	23
5.6	TESTING TECHNIQUES	24
5.6.1	WHITE BOX TESTING	24
5.6.2	BLACK BOX TESTING	25
5.7	SOFTWARE TESTING STRATEGIES	25
5.7.1	VALIDATION TESTING	25
5.7.2	USER ACCEPTANCE TESTING	25
5.7.3	OUTPUT TESTING	25
6	<b>RESULT AND TESTING OF SEDMDroid</b>	26
7	<b>CONCLUSION AND FUTURE ENHANCEMENT</b>	36
	<b>REFERENCES</b>	37
	<b>APPENDIX</b>	40
	<b>PLAGIARISM REPORT</b>	47
	<b>PAPER SUBMISSION STATUS</b>	51



## LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	Architecture Diagram.	12
3.2	ER Diagram	13
3.3	Sequence Diagram	14
3.4	Activity Diagram	15
6.1	Data Selection and Loading	27
6.2	Pre-Processing the Dataset	28
6.3	Data Correlation Plot Graph	29
6.4	Slitting Dataset into Train and Test Data	30
6.5	Feature Reduction	31
6.6	Prediction for Classification of Data	32
6.7	RF-Accuracy Graph	33
6.8	RF ROC Accuracy	34
6.9	XGBOOST Accuracy Graph	35
6.10	XGB ROC Accuracy	36

## LIST OF ABBREVIATIONS

ABBREVIATIONS	ABBREVIATED TO
OS	Operating System
SEDMDroid	Programming model for processing large datasets
AUC/RUC	Area Under Curve/Receiver
OCC	Operating Characteristic Curve
GPS	Global Positioning System
Kaspersky	Security firm
Symantec	Security firm
RF	Random forest
XGBOOST	Extreme Gradient Boosting
SCC	Smart and Connected Communities
PCA	Principal Component Analysis
MaMaDroid	Malware Detection in Mobile Devices
MADAM	Malware Detection and Prevention on Android Malware Devices
ML	Machine Learning
Droid	Android-Based Mobile Operating System

# CHAPTER 1

## INTRODUCTION

SEDMDroid is an enhanced stacking ensemble of deep learning framework for android malware detection.

### 1.1 OVERVIEW

#### **Remote sensing**

Deceptive Android malware is a common issue, with many apps promising to improve user experience, but instead causing harm. For instance, Ads Blocker claimed to eliminate intrusive ads that disrupt phone usage, but instead infected devices with malware that generated even more ads, as reported by cybersecurity experts. The prevalence of such malware can be frustrating for Android users, who are bombarded with unwanted ads that generate revenue for the attackers, even when they're not using the affected app. Additionally, the malware may also produce fake clicks on the ads, further benefiting the perpetrators [11].

#### **Project Introduction**

Cell phones are an extremely essential part of modern living, offering a range of useful features such as email access, web browsing, GPS navigation, and voice recognition. However, with the growing popularity of mobile devices, malware developers have also turned their attention to mobile platforms, aiming to exploit vulnerabilities and cause damage to these devices. For instance, according to Kaspersky's security report, with more than 800,000 new malicious software strains were identified, which was thrice more than the previous year. Symantec also reported an average of one to zero day strike everyday in 2015, along with a significant growth in the volume of malware and many more background also came in the existence in that time period [10]. Researchers and security firms are working together to safeguard mobile devices from such threats and are working hard to create efficient anti-malicious system. Many different techniques are present techniques for the detection of malicious software and analysis, with differing solidity and frailties. Pair of similar methods are static and dynamic analysis,

which involve analysing code to identify malicious behaviour. With billions of users and a large number of independent developers creating mobile apps, several challenges are faced for an end user to assess the reliability of the applications. Additionally, mobile apps have the privilege of accessing delicate instruction, such as connections on the cell phone and Global Positioning System input, after being announced in the file of the surface of android which is getting manifested. This mechanism of authority declaration is critical to support the numerous functionalities of mobile apps, but it also makes smartphones and other Android-powered devices a prime target for cyberattacks. Android, being a free open-source operate software, is right now the most popular smart cell phone platform which has a market part 85 percent and more. The proliferation of Android in smart television, vehicle manoeuvre system, and other smart devices makes it even more appealing to cybercriminals. Therefore, it is essential to continue developing effective anti-malware systems to protect these devices from the growing threat of malware [9].

## **1.2 PROJECT OBJECTIVES**

The primary aim is to detect Android malware efficiently using a dataset. This involves utilising different machine learning algorithms to construct prediction models, as well as assessing their accuracy and performance. The goal is to improve the precision of the classification outcomes.

## **1.3 Problem Statement**

With the increasing usage of Android apps, it has become imperative to detect malicious behaviour in these apps to ensure end-user security and privacy. To address this issue, a ML based structure has been proposed to locate malicious threats in Android apps by classifying them as either malicious or benign.

## **1.4 MOTIVATION**

Protecting user privacy and security: Malware can be used to steal sensitive information from users' devices, including passwords, banking information, and personal data. By detecting and removing malware, android malware detection tools help to protect user privacy and security. Preventing financial loss: Malware can also be used to carry out financial fraud, such as by sending premium-rate text messages or making unauthorized purchases. By detecting and removing malware, android malware detection tools can help prevent financial losses.

Maintaining device performance: Malware can slow down devices and cause crashes, reducing device performance. By detecting and removing malware, android malware detection tools can help to maintain device performance.

.

## **1.5 PURPOSE**

The primary purpose of android malware detection is to identify and remove malicious software (malware) that can harm android devices and steal sensitive user information. Android malware can be introduced to a device through various means, such as malicious apps, email attachments, web downloads, and infected external devices. Once installed on a device, malware can perform various harmful actions, such as stealing user data, displaying unwanted ads, performing unauthorized transactions, and even taking control of the device [13].

Android malware detection tools are designed to scan devices for the presence of malware and remove any threats found. These tools use various techniques to identify malware, such as signature-based detection, behavior-based detection, and machine learning-based detection. The goal of android malware detection is to protect users from the harmful effects of malware and ensure the security and privacy of their personal data.

# **CHAPTER 2**

## **LITERATURE REVIEW**

The goal of this writing survey is to establish the information we need to develop the system and to understand the preexisting systems that have similar workings

### **2.1 Existing System**

In order to combat the rapid proliferation of malicious threats on the devices, an unchanged malevolent finding structure called SEDMDROID is proposed. Such structure follows a twin level structure that incorporates a group which includes base-learners, XGBOOST, and the combination of base-learner return by RF. This framework begins by introducing a dual disruption of attribute room and test room, which make sure the divergence in the preparation subsets, and applies “PCA” separately to each subset, all main modules obtained from “PCA” are kept, also from the entire preparation data set is transformed into an advanced objective, upon which XGBOOST is route to ensure the accuracy of the base learner [15].

#### **2.1.1 Disadvantages of the Existing System**

- A lower learning rate was discovered to be effective in determining whether the executable is benign or malicious.
- The system's performance is significantly poor.
- Low accuracy.

### **2.2 Proposed System**

The target of this suggested model aims to deal with the present method's weaknesses. By employing machine learning algorithms, the model aims to increase the accuracy of detecting malware from Android datasets. Furthermore, it seeks to improve the overall performance of the classification results. The goal is to make the prediction of malware from Android data more reliable and accurate.

### 2.2.1 Advantages of Proposed System

- High Performance
- The system has the ability to handle packed malware and can operate on different types of malware, regardless of the operating system being used.

## 2.3 LITERATURE SURVEY

The proposed model is introduced to overcome all the disadvantages that arise in the existing system. This system will increase the accuracy of the machine learning results by detecting malware from android dataset using machine learning algorithm. It enhances the performance of the overall classification results. Predict the malware from android data is to find the accuracy more reliable [1].

### 2.3.1 Comprehensive Review of Recent Literature for Detecting Malware on Android Devices

**Title:** Coevolution of Mobile Malware and Anti-Malware

**Year:** 2018

**Author:** Sevil Sen, Emre Aydogan, Ahmet I. Aysan

#### **Methodology**

Cell Phone malicious software poses a serious threat to computing device confidentiality, with latest malicious software on a daily basis as if it introduces new security risks. Current safety measures are normally designed safeguard handheld devices towards recognised dangers, However, they are subject to unexpected hazards. This research looked at authors who investigate the application of evolving computational methods to develop additional mobile threat strains that can flourishingly evade statistical analysis based malware protection solutions, as well as automatically creating stronger security measures towards them. The co-evolutionary A viable possibility for constructing a more resilient structure to new threats and for system testing is the arms race process. This study represents first application for co-evolutionary computation to address this problem, offering a promising approach to addressing the challenges posed by evolving mobile malware. Such a study presents an efficient way of detecting malware on the android devices which may cause problems in the future [2].

### **Advantage**

□ One key advantage of this system over previous ones is that it can automatically detect significant features without requiring any human input or oversight.

### **Disadvantage**

□ The functioning of computers and computer networks can be disrupted by intentional or unintentional causes, such as catastrophic breakdowns or significant performance declines of an individual computer or network

**Title:** Internet of Things and Big Data Analytics for Smart and Connected Communities

**Year:** 2015

**Author:** YUNCHUAN SUN, HOUBING SONG, ANTONIO J. JARA AND RONGFANG BIE

### **Methodology**

The notion of "SCC", which expands on the idea about smart towns, is discussed in this study. "SCC's" mission is to solve a community's conservation, regeneration, livability, and attainability issues. According to the article, Internet of Things (IoT) technology can offers a system of interconnected gadgets and advanced sensors for SCC, and big data analytics can enable the needed immediate oversight for "SCC"[3]. As significant Internet of Things (IoT) innovations for fostering "SCC", the authors highlight cellular audience detection and cyber physical cloud-based computing. The article additionally contains a case study of TreSight, an Internet of Things and big data analytics solution for innovative tourism and sustaining heritage preservation in "Trento".

### **Advantage**

□ One notable difference between mobile devices and mote-class sensors is that mobile devices typically have much greater storage, computation, and communication resources, as well as built-in multimodal sensing capabilities.

### **Disadvantage**

□ Ensuring the security and privacy of data collected and transmitted by IoT devices is a challenging task, especially with their growing popularity and usage. It can be difficult to investigate as the popularity has become more enthusiastic.



**Title:** MalPat: Mining Patterns of Malicious and Benign Android Apps via Permission-Related APIs

**Year:** 2017

**Author:** Guanhong Tao, Zibin Zheng, Ziyang Guo, and Michael R. Lyu

### **Methodology**

It focuses on the rise of Android application marketplaces and the corresponding increase in Android malware. The authors offer “MalPat”, a computerised identification of malware mechanism, which mines virus patterns that are disguised as well as extracts API that are highly vulnerable that are commonly utilised in malicious apps [4]. This approach is different from existing detection tools that depend on the individual setting of functionality sets according to authorizations, critical assets, motives, etc. The goal of MalPat is to differentiate between malignant and dangerous Android applications, and help Unknown harmful programmes are addressed in mobile application markets.

### **Advantage**

□ To obtain sensitive data that pertains to user privacy, some malicious apps on Android pretend to offer the same functions as legitimate apps

### **Disadvantage**

□ By default, Android grants "normal" permissions to apps, which include accessing the internet, without requiring user approval. These permissions are deemed safe and shouldn't compromise your privacy or your device's performance. However, for more sensitive actions, such as accessing contacts or using the camera, Android requires the user's explicit consent by requesting "dangerous" permissions.

**Title:** EveDroid: Event-Aware Android Malware Detection Against Model Degrading for IoT Devices

**Year:** 2019

**Author:** Tao Lei, Zhan Qin, Zhibo Wang, Qi Li, and Dengpan Ye

### **Methodology**

The article introduced EveDroid, a structure which helps in detecting Malicious threats on Android based Internet of Things gadgets, with the goal for keeping the IoT environment safe by reducing the presence of malicious software. To accomplish this, EveDroid employs an

event grouping method that is robust against API changes and capable of capturing hidden behaviors of Android apps. A neural network is then utilized obtaining advanced interpretations from these occurrence groups as well as produce predictions [5].

### **Advantage**

□ To assess the efficiency of hyperparameters in our model, we conducted an evaluation. We also conducted a comparative analysis between EveDroid and different malicious software recognition systems, such as MaMaDroid, which demonstrate a superiority within our structure.

### **Disadvantage**

□ Android is a target because app distribution is easier and there are so many Android devices around the world.

**Title:** Toward Engineering a Secure Android Ecosystem: A Survey of Existing Techniques

**Year:** 2016

**Author:** MENG XU, CHENGYU SONG, YANG JI, MING-WEI SHIH, KANGJIE LU, CONG ZHENG, RUIAN DUAN, YEONGJIN JANG, BYOUNGYOUNG LEE, CHENXIONG QIAN, SANGHO LEE, and TAESOO KIM

### **Methodology**

The accessibility and adaptability It has become an increasingly common operating system for portable electronics. an attractive target for attacks. While many solutions have been proposed to address specific security issues, the increasing complexity of these problems calls for a systematic re-evaluation of the platform's security architecture and practices. To that goal,, this document provides a thorough examination review for recent research on Android security, focusing on There are a pair of technological layers: platforms and environments. We discuss the constraints of existing answers and indicate unresolved issues for further investigation. Drawing according to this information, we propose a roadmap for the future-oriented Droid environment that prioritizes safety as well as aims to mitigate the risks of malicious attacks [6].

### **Advantage**

□ TaintDroid's ability to operate on real devices, providing real-time monitoring of the hardware and sensors, has made it a popular choice for Android app behavior analysis. Thus it can be better and easier way to analyse the malware for Android devices.

### **Disadvantage**

□ It has limitations due to its reliance on human requirements, that are susceptible to inaccuracies and rigid software modifications.

**Title:** MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention

**Year:** 2016

**Author:** Andrea Saracino, Daniele Sgandurra, Gianluca Dini and Fabio Martinelli

### **Methodology**

The growing the quantity of harmful programmes, collectively known as malicious threat, poses a significant warning to the security, finances, and coherence of Android users' devices and files. By analyzing their behavior, however, we can classify malware into distinct behavioral classes that perform specific misbehaviors. These misbehaviors can be monitored across different levels of the Android system to identify and categorize them. This paper presents “MADAM” is a hosted security solution for Android smartphones that examines and correlates data at 4 different system stages: the kernel, usage, individual, and module. By considering the characteristics of real-world malware, MADAM is designed to detect and prevent almost all malicious behaviors. The system employs two simultaneous algorithms and an indicator based on cognitive signals, and is able to effectively block over 96% of malicious apps from three large datasets totaling about 2,800 apps [7].

### **Advantage**

□ One advantage of mobile devices is that they eliminate the need for deploying large-scale wireless sensor networks, which can be costly and time-consuming, as mobile devices are carried by people wherever they go and whatever they do.

### **.Disadvantage**

□ "By leveraging multi-level deep learning, it becomes possible to model and capture the intricate distribution of malware data."

**Title:** SherLock vs Moriarty: A Smartphone Dataset for Cybersecurity Research

**Year:** 2014

**Author:** Yisroel Mirsky, Asaf Shabtai, Lior Rokach, Bracha Shapira and Yuval Elovici

### **Methodology**

To showcase potential applications within the information set, We performed a simple malware evaluation based on the obvious classifications provided and illustrated how the “SherLock dataset” could test methods for ongoing identification of users. An urge to researchers and safety experts can use this information for study and improvement objectives. Going forward, we intend to broaden the study of incorporating Additional participants and Moriarty modifications. With the information, along side the current data Getting together, there will be a wider range of chances for scientific for researchers [8].

**Advantage**

□ One effective approach to manage a vast number of period information is by reducing the quality of joining procedures needed when generating datasets in the final database.

**.Disadvantage**

□ "Misconfigured firewalls have the potential to restrict user actions on the internet until the firewall settings are properly adjusted."

## **CHAPTER 3**

# **SYSTEM ARCHITECTURE AND DESIGN OF SEDMDroid**

### **3.1 ARCHITECTURE DIAGRAM**

The practise of picking data that predicts Android malware is known as data selection. This dataset contains details about the permissions and apps that Android malware uses. The act of deleting unnecessary data from a dataset is known as "data pre-processing". Removal of missing data and encoding of categorical data. Missing data removal: Using an imputer library, null values, such as missing values, are eliminated in this procedure.

Categorical data encoding: Variables with a limited number of label values are what are meant by categorical data. that the vast majority of machine learning algorithms want numerical input and output variables. Categorical data is converted to integer data using one integer and one hot encoding.

Data separation is the division of available data into groups. Two sections are typically used for cross-validation. A predictive model is created using a subset of the data, and the other is used to assess the model's effectiveness. Analysing data mining models requires dividing the data into training and testing sets. The majority of the data is often used for training, and a lesser amount is used for testing when you divide a data set into a training set and a testing set. Shrinking of features A set of correlated variables is transformed into a set of uncorrelated variables using an orthogonal transformation as part of the statistical method known as principal component analysis (PCA). The most often used tool in machine learning for predictive models and exploratory data analysis is PCA, In fig. 1 we can see the architecture diagram showcasing the software components of the model.

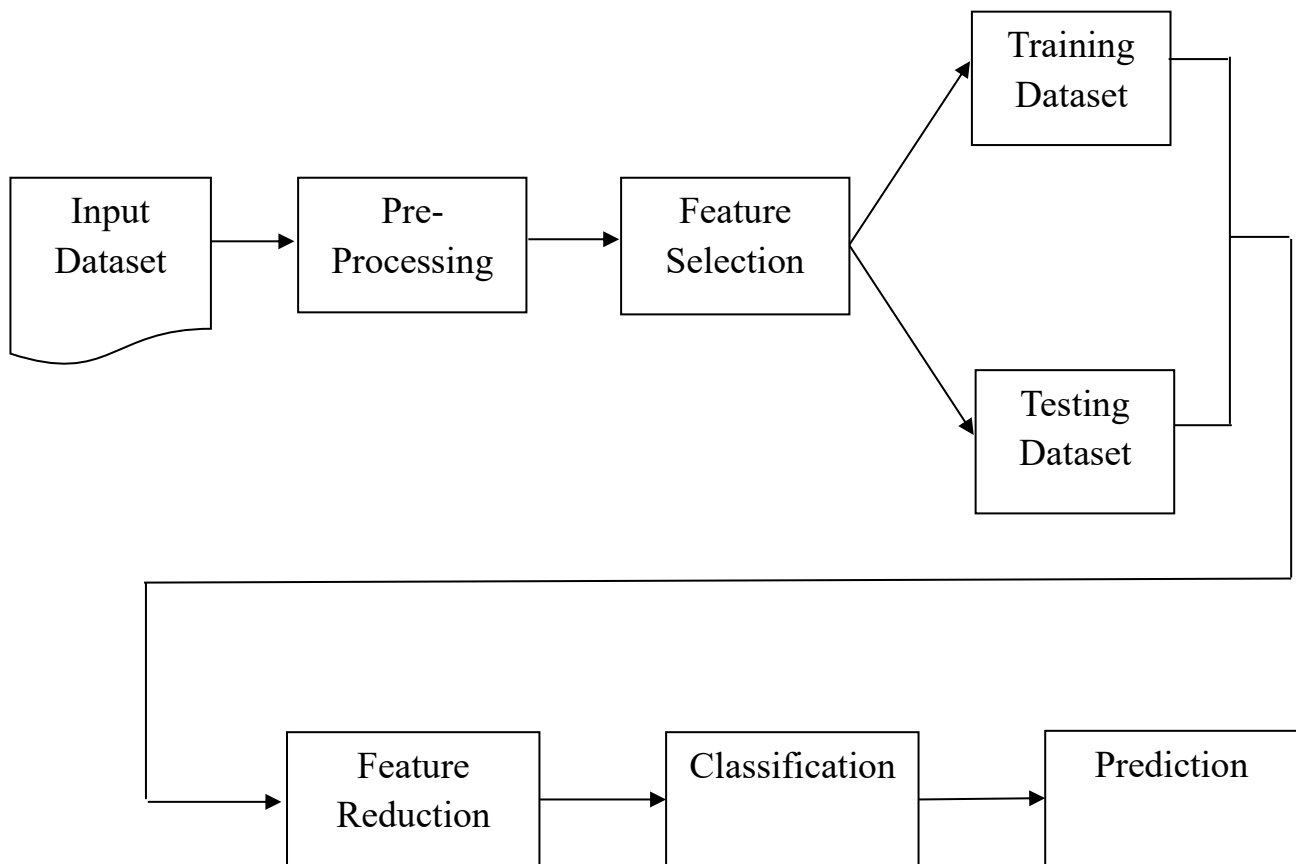


Figure 3.1 Architecture Diagram

### 3.2 ER DIAGRAM

Encoding for categorical data: categorical data refers to variables having a small number of label values. Most machine learning algorithms prefer numerical input and output variables. Using one hot encoding and one integer, categorical data is transformed into integer data.

The partitioning of available data into groups is known as data separation. Cross-validation normally involves two sections. A portion of the data is used to develop a predictive model, while the remainder is used to evaluate the model's performance. Data must be separated into training and testing sets before data mining models can be analysed. When you split a data set into a training set and a testing set, the bulk of the data is frequently used for training and the rest for testing.

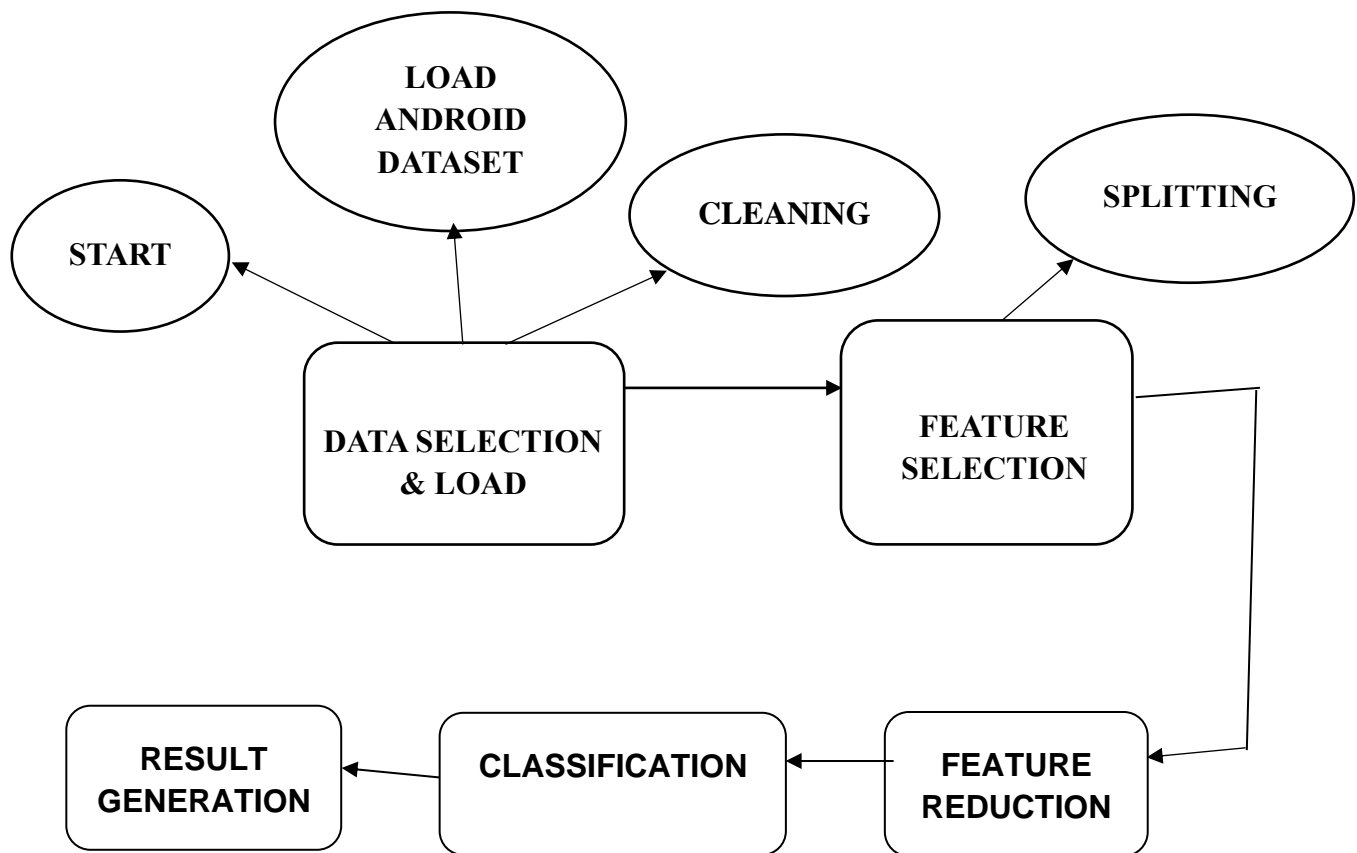


Figure 3.2 ER Diagram

For classification, regression, and other tasks, random forests or random decision forests are ensemble learning techniques that build a large number of decision trees during the training phase and output the class that represents the mean of the classes or the mean or average prediction (for regression) of the individual trees.

### 3.3 SEQUENCE DIAGRAM

The habit of selecting data that indicates an Android virus is called "data selection. The apps and permissions that Android malware employs are described in this dataset. The term "data pre-processing" refers to the process of removing extraneous data from a dataset. Encoding of categorical data and the removal of missing data Missing data removal: In this process, null values, such as missing values, are removed using an imputer library. Encoding for categorical data: The vast majority of machine learning algorithms prefer numerical input and output

variables; hence, categorical data refers to variables with a constrained set of label values. Using one integer and one hot encoding, categorical data is transformed to integer data.

Encoding for categorical data: Variables with a limited number of label values are referred to as categorical data. Numeric input and output variables are preferred by the majority of machine learning algorithms. Categorical data is converted to integer data using one hot encoding and one integer. Data separation is the division of available data into groups. Two portions are often used in cross-validation. The creation of a predictive model uses a subset of the data, and the effectiveness of the model is assessed using the remaining data. Before data mining models can be examined, data must be divided into training and testing sets. The majority of the data is typically used for training and the remaining portion for testing when a data set is divided into a training set and a testing set.

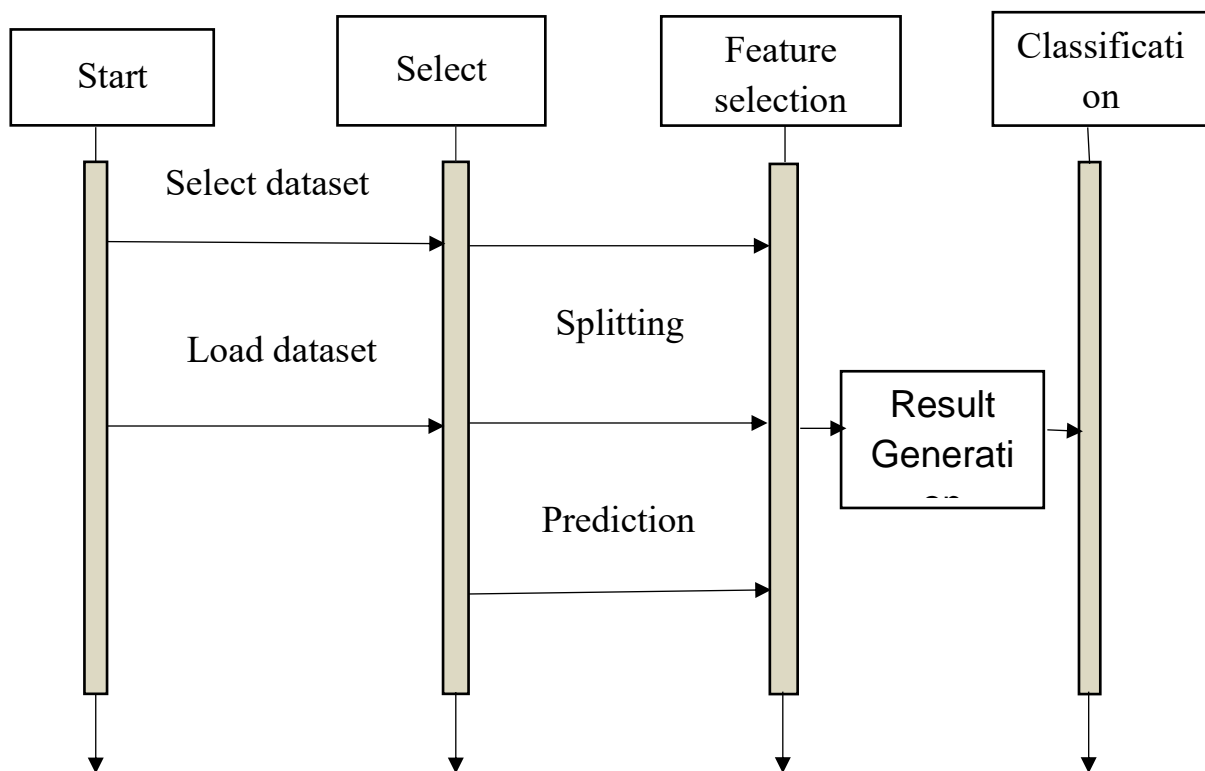


Figure 3.3 Sequence Diagram

### 3.4 ACTIVITY DIAGRAM

Android malware detection is a process of identifying malicious applications designed to target the Android operating system. The detection process involves several steps, including input data collection, pre-processing, feature selection, and dataset creation for training and testing.



The first step in the detection process is input data collection, where data is gathered from various sources such as app stores, user feedback, and security researchers. This data is then used to create a dataset for malware detection. The next step is pre-processing, where the collected data is cleaned and transformed into a format that can be used for analysis.

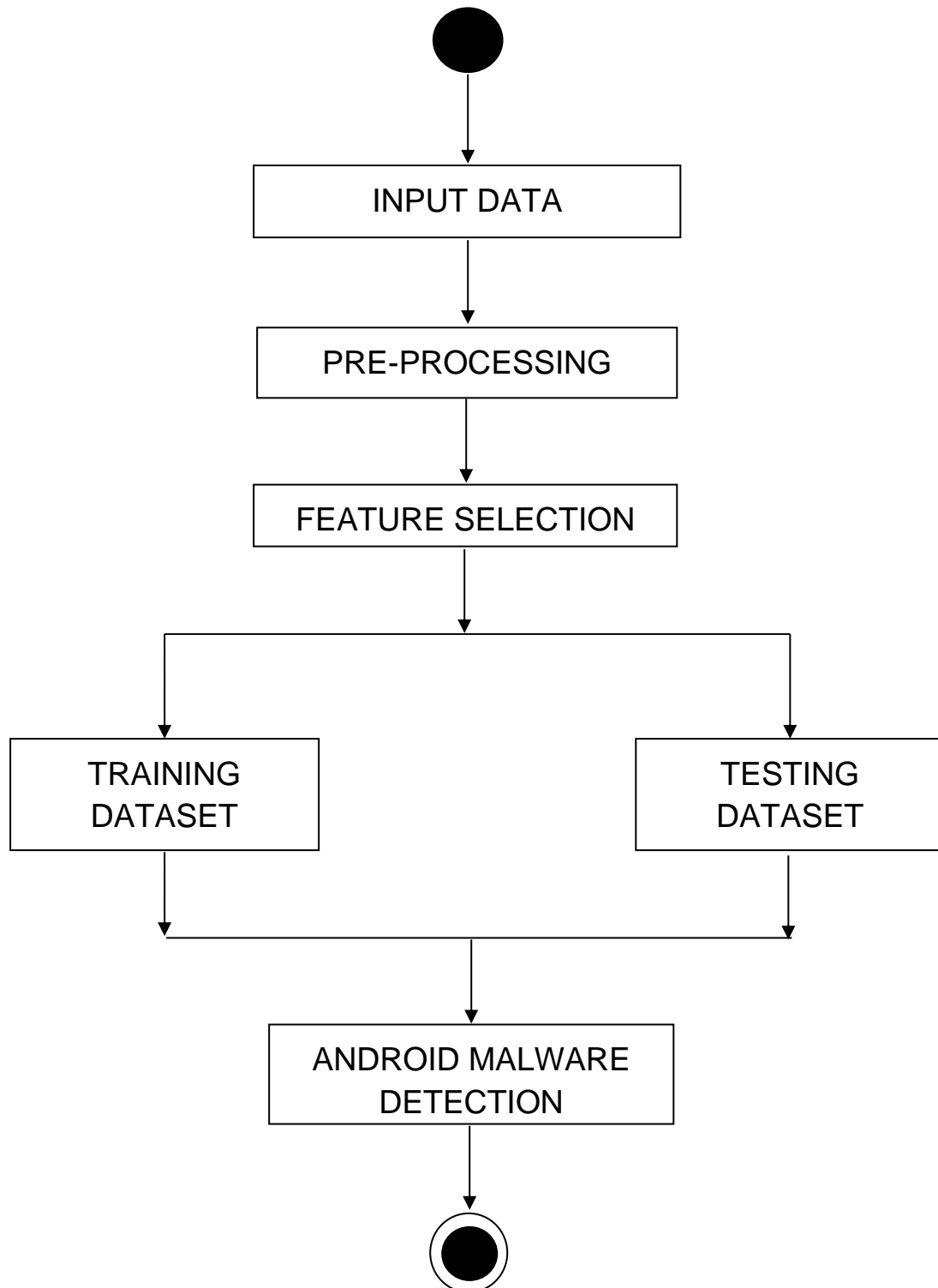


Figure 3.4 Activity Diagram

Feature selection is the process of selecting relevant features or attributes from the pre-processed dataset. This is important to reduce the size of the dataset and improve the accuracy of the detection process. The selected features are used to train machine learning models that can identify malware based on the patterns and characteristics of the selected feature

The training dataset is used to train the machine learning models, and the testing dataset is used to evaluate the performance of the trained models. The models are evaluated based on metrics such as accuracy, precision, recall, and F1-score.

In conclusion, android malware detection involves several steps, including input data collection, pre-processing, feature selection, and dataset creation for training and testing. The detection process uses machine learning models that are trained on selected features from the dataset and evaluated on testing data to identify malicious applications targeting the Android operating system.

# **CHAPTER 4**

## **IMPLEMENTATION OF SEDMDroid**

### **4.1 MODULES**

The following actions must be taken in order to construct the system:

- Data Selection and Loading
- Data Preprocessing
- Feature Selection
- Feature Reduction
- Classification
- Prediction
- Result Generation.

### **4.2 MODULE DESCRIPTION**

- The technique of picking information for predicting Android malware involves carefully choosing relevant information from a dataset.
- The dataset typically includes information about Android malware apps and the permissions they request..

.

#### **4.2.1 DATA SELECTION AND LOADING**

- The data selection is the process of selecting the data predicting Android Malware.
- The dataset which contains the information about android malware apps and permissions.

#### **4.2.2 DATA PREPROCESSING**

- Data pre-processing aims to remove unwanted data from the dataset and prepare it for analysis. This involves two steps:..

- **Missing data removal** Removing missing data: Null values and missing data are eliminated by utilising the impetus library.
- **Categories of information representation** Statistics with classifications consists of parameters that have a limited number of labelled values. Given that the majority of ML methods demand quantitative parameters for both input and output,, categorical data must be converted to integer data using techniques such as integer and one-hot encoding..

#### **4.2.3 SPLITTING DATASET INTO TRAIN AND TEST DATA**

- Information division refers too as process in which dividing the present dataset the two parts as ‘cross validation purposes’. first part of the dataset is employed to create a representation for predicting, while the second part is used for assessing the performance of the design.
- The separation of dataset is a crucial step in evaluating data mining models. Normally, the majority of the information is allocated fA lesser amount is allotted to evaluation and the remainder for instruction.

#### **4.2.4 REDUCTION FEATURE**

“PCA” is an analytical technique which involves a reversible conversion of a creating a set of unrelated variables from a set of linked ones. It is a commonly utilised instrument in ML and data exploration to create models that anticipate.

### **4.3 CLASSIFICATION OF THE DATASET**

#### **4.3.1 RANDOM FOREST**

Random forests are a popular group instruction technique used in a variety of tasks including “classification” and “regression”. During training, the method builds multiple decision trees and outputs the class that is most frequently predicted across all of the trees (in classification tasks) or the mean/average prediction across all trees (in regression tasks) [16].

#### **4.3.2 XGBOOST**

XGBoost is a popular ensemble learning algorithm for ML that utilises decision trees and uses a system for improving variations. While synthetic neural systems are mostly the top-performing method for prediction problems involving unstructured data, At present, decision tree-based techniques are being thought of the best-in-class approach for minimal to moderately sized structured or data tables [17].

#### **4.4 PREDICTION**

- An important goal of this project is to predict the presence of Android malware using a dataset.
- By the application of several approaches and methods, this project aims to improve the accuracy and overall implementation of the forecasted outcomes.

#### **4.5 OUTCOME GENERATION**

The end output of the proposed approach is determined through a comprehensive characterization and forecasting process. The effectiveness of the model is concluded based on several outcomes, including precision, roc precision, and many more.

- Precision
- ROC Accuracy
- Recall
- F1-Measure
- Sensitivity
- Specificity
- Matthews Correlation Coefficient

## CHAPTER 5

### SYSTEM REQUIREMENTS OF SEDMDroid

#### 5.1 HARDWARE REQUIREMENT

- System : Pentium IV 2.4 GHz
- Hard DisK :200 GB
- Mouse: Logitech.
- Keyboard:110 keys enhanced
- RAM: 4GB

#### 5.2 SOTWARE REQUIREMENT

- O/S : Windows 7.
- Language : Python
- Front End : Anaconda Navigator – Spyder

#### 5.3 SOTWARE DESCRIPTION

##### Python

Python is one of those rare languages which can claim to be both simple and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

##### Features

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one

of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

### **Easy to Learn**

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

### **Free and Open Source**

Python is an example of a FLOSS (Free/Libre and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

### **High-level Language**

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

### **Portable**

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, and # -\*- coding: utf-8 -\*-

z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like Kivy to create games for your computer and for iPhone, iPad, and Android.

### **Interpreted**

This requires a bit of explanation.

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called byte codes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

### **Object Oriented**

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

### **Extensible**

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

### **Embeddable**

You can embed Python within your C/C++ programs to give scripting capabilities for your program's users.

### **Extensive Libraries**

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the Batteries Included philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the Python Package Index.

## **5.4 FEASIBILITY STUDY**



The feasibility study is carried out to test whether the proposed system is worth being implemented. The proposed system will be selected if it is best enough in meeting the performance requirements.

The feasibility carried out mainly in three sections namely [19].

- Economic Feasibility
- Technical Feasibility
- Behavioural Feasibility

#### **5.4.1 Economic Feasibility**

Economic analysis is the most frequently used method for evaluating effectiveness of the proposed system. More commonly known as cost benefit analysis. This procedure determines the benefits and saving that are expected from the system of the proposed system. The hardware in system department is sufficient for system development [20].

#### **5.4.2 Technical Feasibility**

This study centre around the system's department hardware, software and to what extent it can support the proposed system department is having the required hardware and software there is no question of increasing the cost of implementing the proposed system. The criteria, the proposed system is technically feasible and the proposed system can be developed with the existing facility.

#### **5.4.3 Behavioral Feasibility**

People are inherently resistant to change and need sufficient amount of training, which would result in lot of expenditure for the organization. The proposed system can generate reports with day-to-day information immediately at the user's request, instead of getting a report, which doesn't contain much detail.

### **5.5 TESTING OF PRODUCTS**

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. . A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways [7]. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that “all gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

### **5.5.1 Unit Testing**

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’.

The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module [1]. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

### **5.5.2 Integration Testing**

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data [3]. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

- i) Top-down integration testing.
- ii) Bottom-up integration testing.

## **5.6 TESTING TECHNIQUES/STRATEGIES**

### **5.6.1 WHITE BOX TESTING**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we  
Derived test cases that guarantee that all independent paths within a module have been exercised at least once.

### **5.6.2 BLACK BOX TESTING**

- Black box testing is done to find incorrect or missing function
- Interface error
- Errors in external database access
- Performance errors.
- Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’. It tests the external behaviour of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

## **5.7 SOFTWARE TESTING STRATEGIES**

### **5.7.1 Validation Testing**

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many, But a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer.

### **5.7.2 User Acceptance Testing**

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

### **5.7.3 Output Testing**

After performing the validation testing, the next step is output asking the user about the format

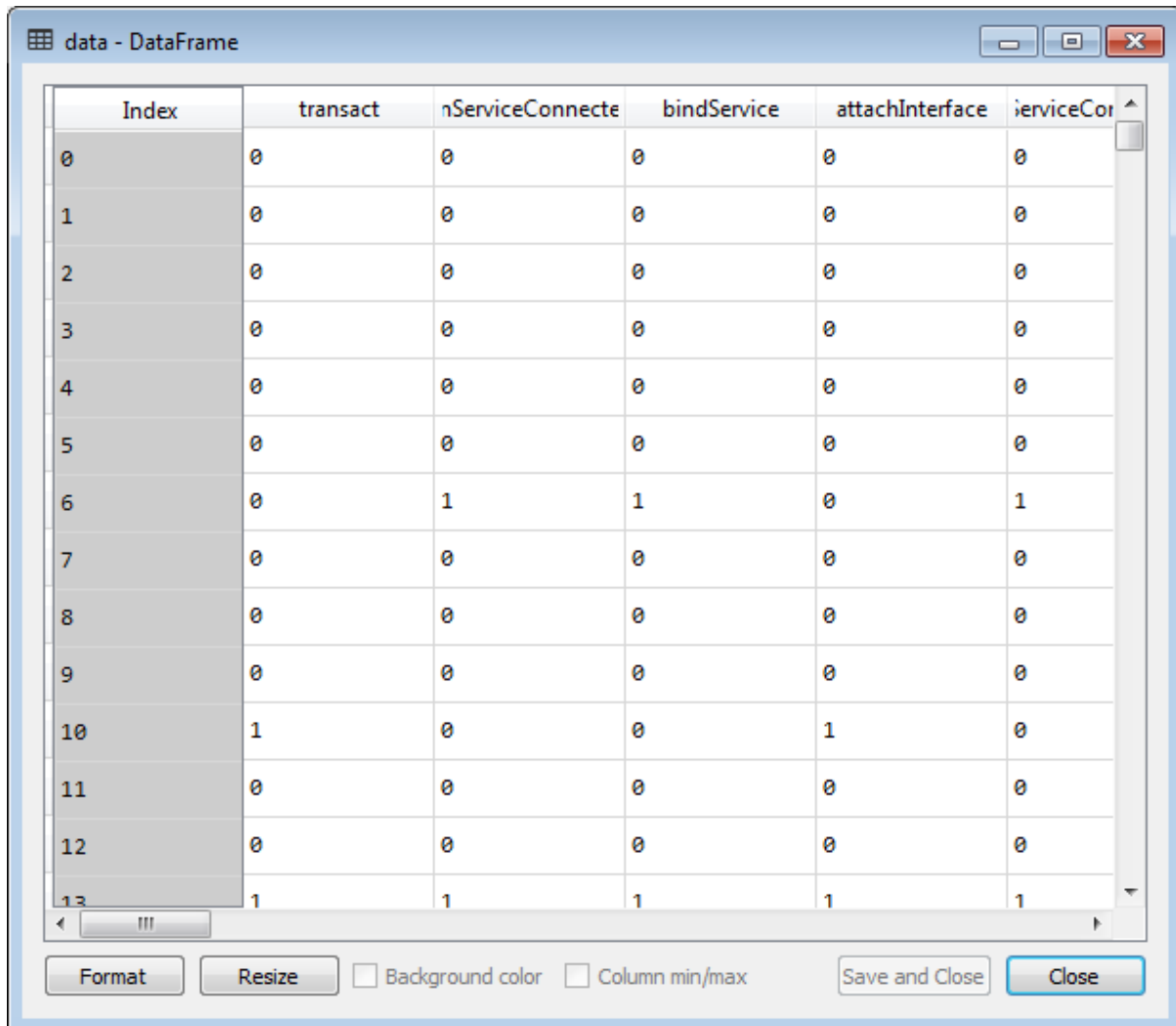
required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Thus no connection in the system is seen.

## CHAPTER 6

### RESULTS AND TESTING OF SEDMDroid

In addition to the steps mentioned above, it is important to note that the quality and quantity of the input data can have a significant impact on the accuracy and effectiveness of the Android malware detection system.

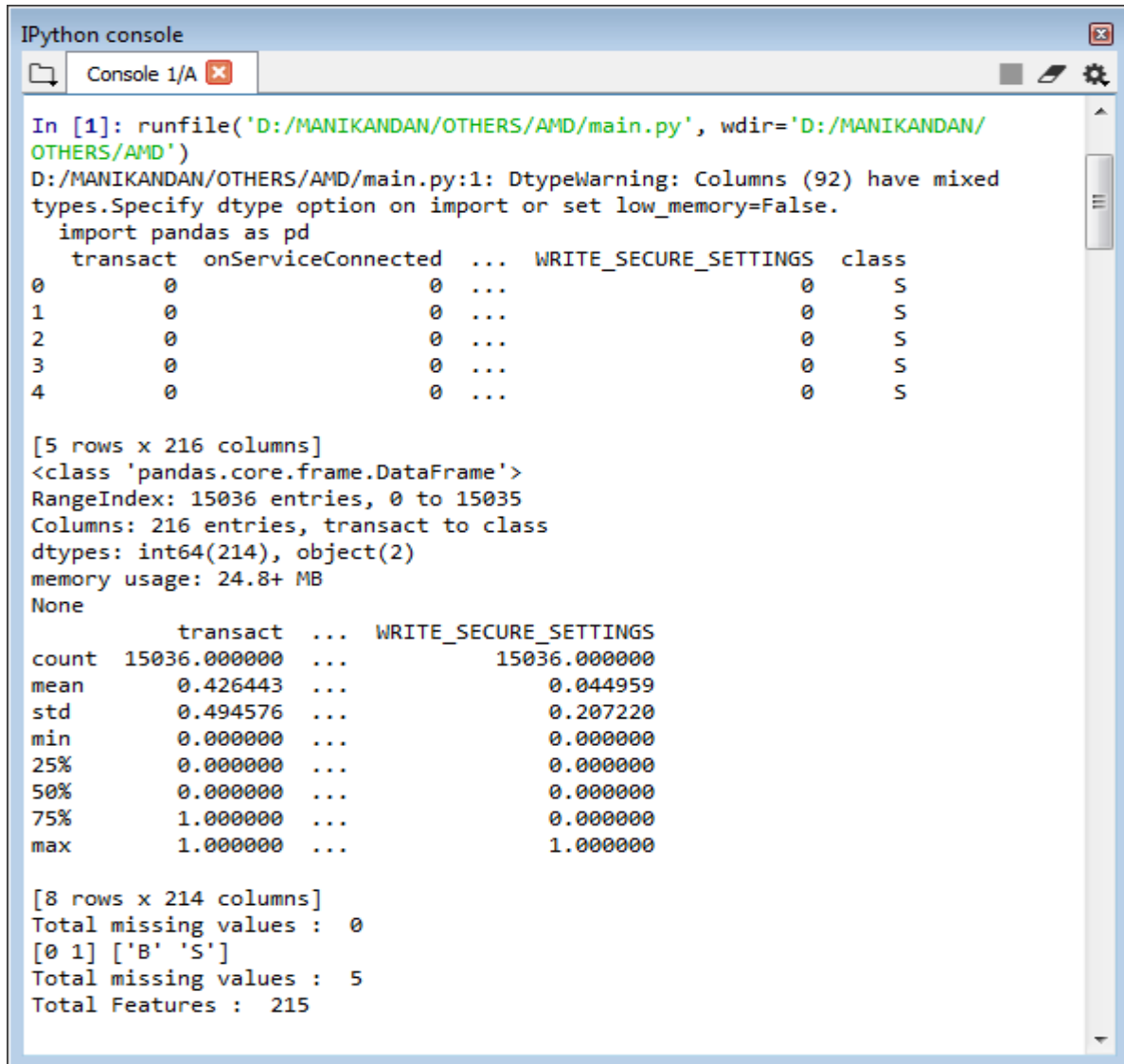
It is essential to ensure that the input data is relevant, reliable, and up-to-date. Additionally, pre-processing techniques such as data cleaning, normalization, and scaling can further improve the quality of the input data.



Index	transact	nServiceConnecte	bindService	attachInterface	serviceCor
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	1	1	0	1
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0
10	1	0	0	1	0
11	0	0	0	0	0
12	0	0	0	0	0
13	1	1	1	1	1

Figure 6.1 Data Selection and Loading

Another important step in the feature selection process is to identify and eliminate redundant or irrelevant features that may negatively impact the performance of the prediction model. This can be achieved through techniques such as correlation analysis, principal component analysis, and mutual information gain.



```

IPython console
Console 1/A

In [1]: runfile('D:/MANIKANDAN/OTHERS/AMD/main.py', wdir='D:/MANIKANDAN/OTHERS/AMD')
D:/MANIKANDAN/OTHERS/AMD/main.py:1: DtypeWarning: Columns (92) have mixed
types.Specify dtype option on import or set low_memory=False.
import pandas as pd
  transact  onServiceConnected  ...  WRITE_SECURE_SETTINGS  class
0          0                  0  ...                  0          S
1          0                  0  ...                  0          S
2          0                  0  ...                  0          S
3          0                  0  ...                  0          S
4          0                  0  ...                  0          S

[5 rows x 216 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15036 entries, 0 to 15035
Columns: 216 entries, transact to class
dtypes: int64(214), object(2)
memory usage: 24.8+ MB
None
      transact  ...  WRITE_SECURE_SETTINGS
count  15036.000000  ...      15036.000000
mean      0.426443  ...      0.044959
std      0.494576  ...      0.207220
min      0.000000  ...      0.000000
25%      0.000000  ...      0.000000
50%      0.000000  ...      0.000000
75%      1.000000  ...      0.000000
max      1.000000  ...      1.000000

[8 rows x 214 columns]
Total missing values : 0
[0 1] ['B' 'S']
Total missing values : 5
Total Features : 215

```

Figure 6.2 Pre-Processing the Dataset

The first step in data preprocessing is to extract the features from the files. This is done by running the SEDMDroid tool on the files and generating a feature vector for each app. The feature vector is a list of numerical values that represent the features extracted from the file which is acting as the malware for the android device.

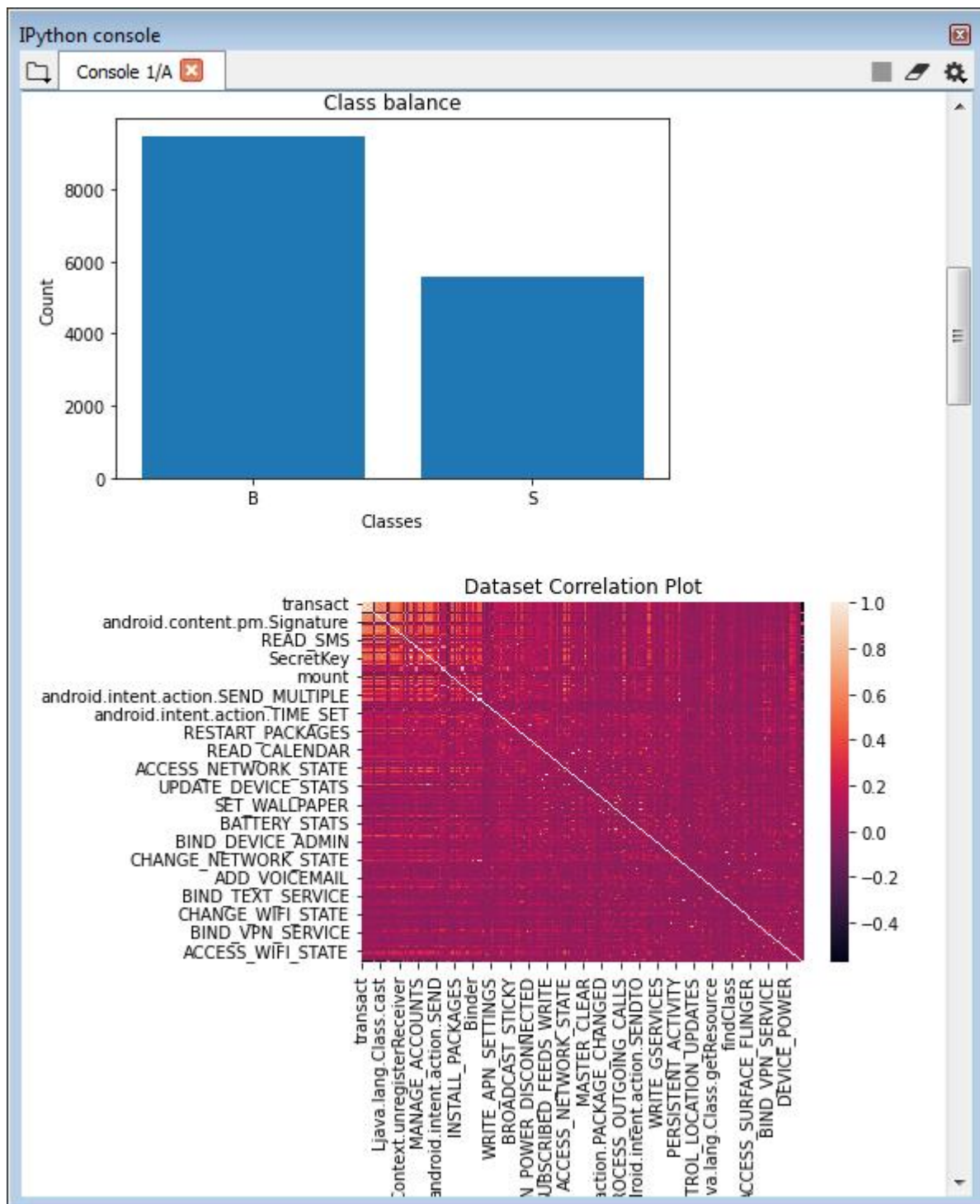


Figure 6.3 Data Correlation Plot Graph

Furthermore, it is critical to evaluate the performance of the Android malware detection system using appropriate performance metrics such as ROC curves, precision-recall curves, and confusion matrices. These metrics can help identify potential issues with the prediction model and guide the refinement of the system.

	0	1	2	3	4
0	1	1	1	1	1
1	0	1	1	0	1
2	1	0	0	1	0
3	0	1	1	0	1
4	1	1	1	1	1
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	1	1	1	1	1
9	0	0	0	0	0
10	0	0	0	0	0
11	1	1	1	1	1

Figure 6.4 Splitting Dataset into Train and Test Data

Data splitting is the act of partitioning available data into two portions, usually for cross-validatory purposes. One portion of the data is used to develop a predictive model, and the other to evaluate the model's performance. Separating data into training and testing sets is an important part of evaluating data mining models. Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing.

PCA is a statistical method used in data analysis and machine learning to reduce the complexity of a large dataset while retaining the most relevant information. It is an unsupervised learning method that transforms the data into a new coordinate system, where the first principal component explains the maximum variance in the dataset and each subsequent component



explains as much of the remaining variance as possible.

PCA is particularly useful in data exploration, as it helps to identify patterns and relationships among the variables. It can also be used to eliminate noise and redundancy in the data, resulting in more efficient and accurate models.

In the context of Android malware detection using Sedroid, PCA can be used as a feature reduction technique to identify the most significant features that contribute to the classification of malware. This can help to reduce the dimensionality of the data and improve the performance of the classification models. PCA can also be used to identify the correlations among the features and eliminate the redundant ones, thus improving the interpretability of the models.



Figure 6.5 Feature Reduction

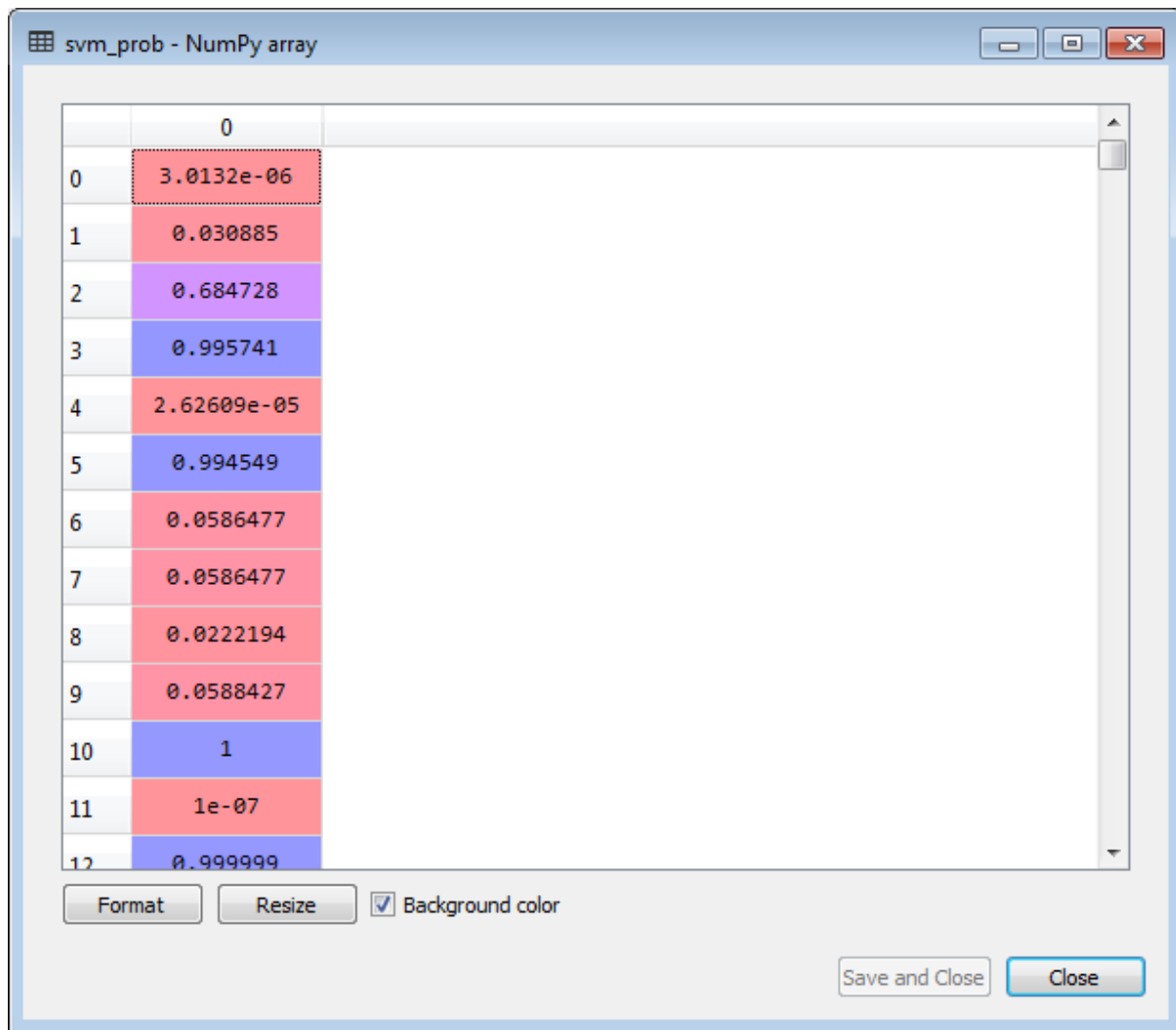


Figure 6.6 Prediction for Classification of Data

Finally, it is important to note that the Android malware detection system should be regularly updated and retrained using the latest data to ensure its continued effectiveness and accuracy. This can be achieved through the use of machine learning algorithms such as supervised learning, unsupervised learning, and reinforcement learning.

Random forests are a popular group instruction technique used in a variety of tasks including “classification” and “regression”. During training, the method builds multiple decision trees and outputs the class that is most frequently predicted across all of the trees (in classification tasks) or the mean/average prediction across all trees (in regression tasks).

The RF algorithm works by creating multiple decision trees using different subsets of the training dataset. Each tree in the RF model uses a random subset of features to split the data

at each node. This process is repeated for each tree until a set number of trees is reached. Once all the trees are created, the final prediction is made by taking the majority vote from all the individual trees.

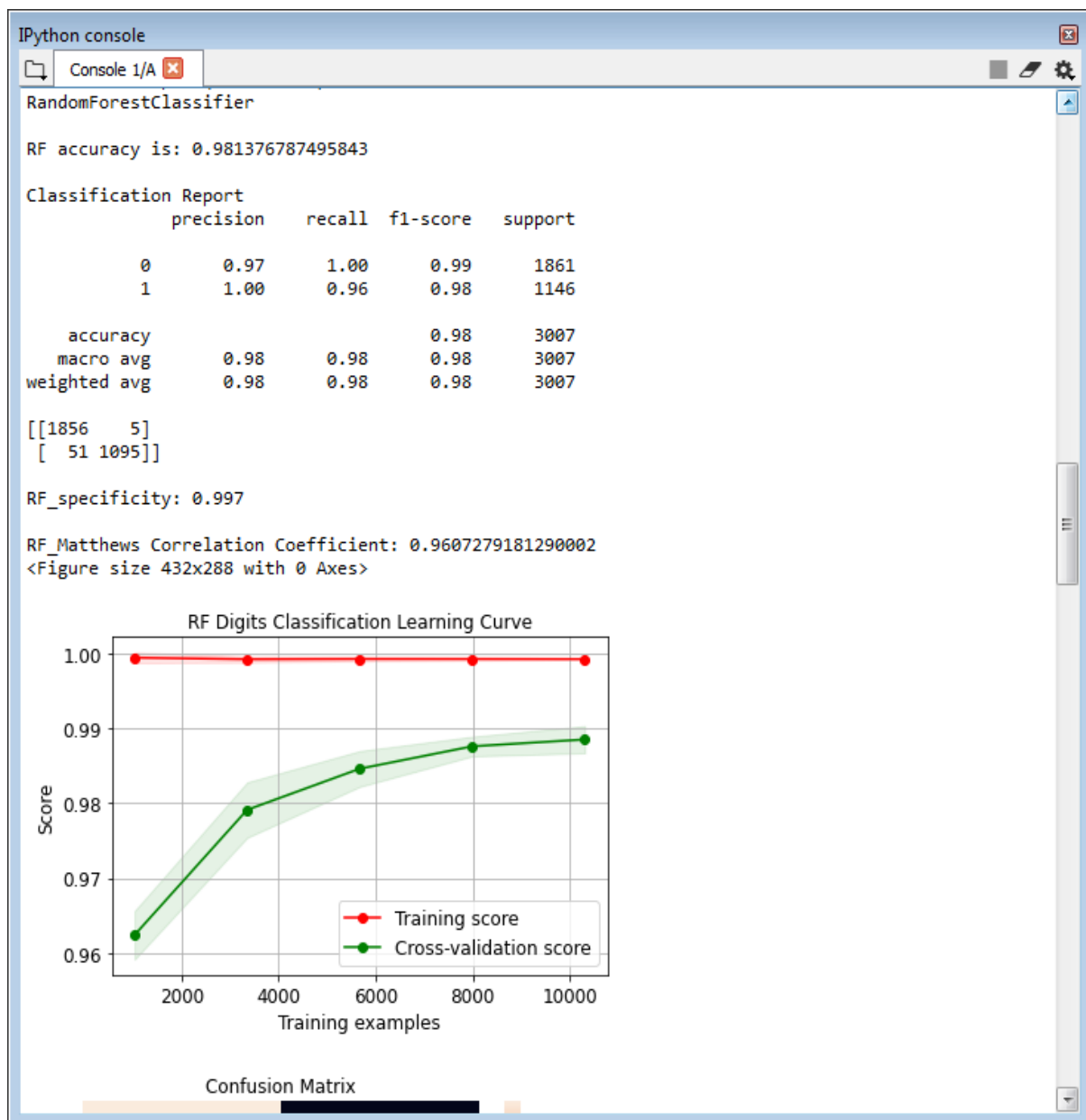


Figure.6.7 RF Accuracy Graph

One of the advantages of using RF classifier for Android malware detection is its ability to handle a large number of features. Since APK files contain a large amount of data, it is important to extract relevant features for accurate classification. RF classifier can handle both numerical and categorical features, which makes it a suitable algorithm for this task.

Moreover, RF classifier can also provide feature importance scores that help in identifying the most important features for classification. This information can be used to further optimize the feature selection process and improve the accuracy of the model.

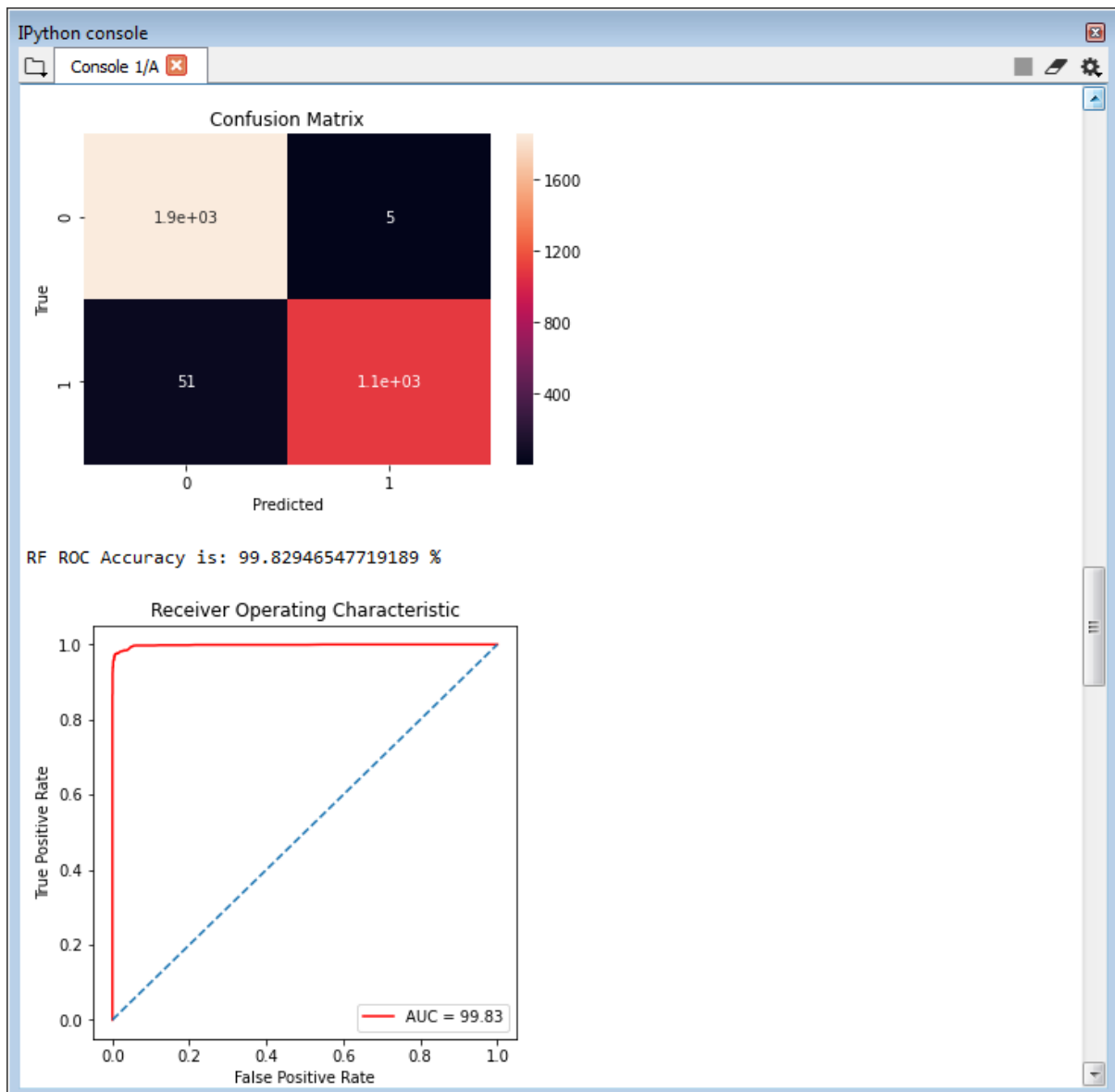


Figure 6.8 RF ROC Accuracy

XGBoost is a popular ensemble learning algorithm for ML that utilises decision trees and uses a system for improving variations. While synthetic neural systems are mostly the top-performing method for prediction problems involving unstructured data, At present, decision tree-based techniques are being thought of the best-in-class approach for minimal to moderately sized structured or data tables.

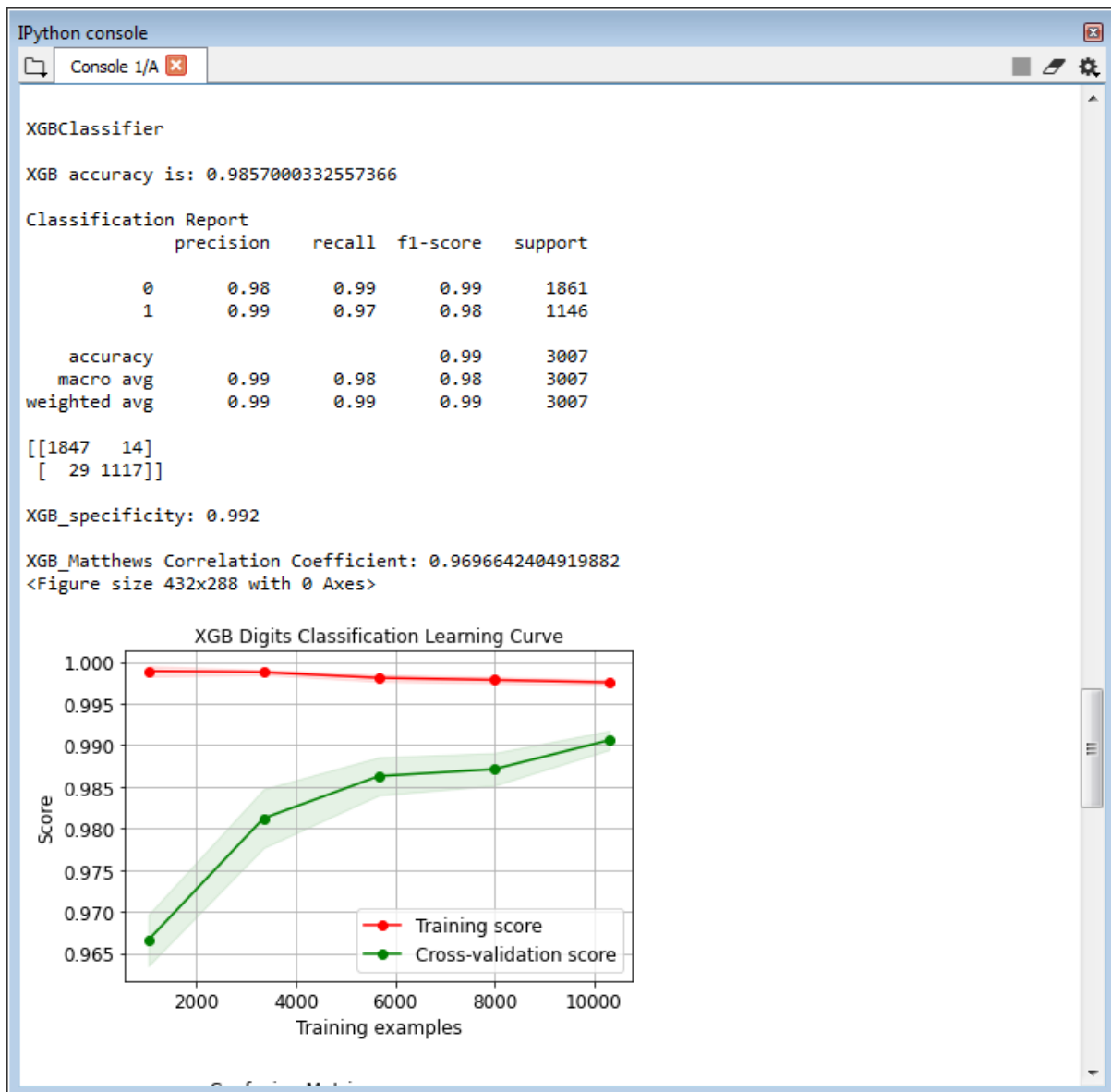


Figure 6.9 XGBOOST Accuracy Graph

When building an XGBoost model for Android malware detection, the first step is to preprocess the data, which involves removing any irrelevant or redundant features and transforming the remaining data into a suitable format for the model. This is followed by splitting the data into training and testing sets and using the training set to train the XGBoost model.

The trained model is then evaluated on the testing set, using measures such as accuracy, precision, recall, and F1 score, to assess its performance. XGBoost also allows for tuning of hyperparameters to improve the model's accuracy further.

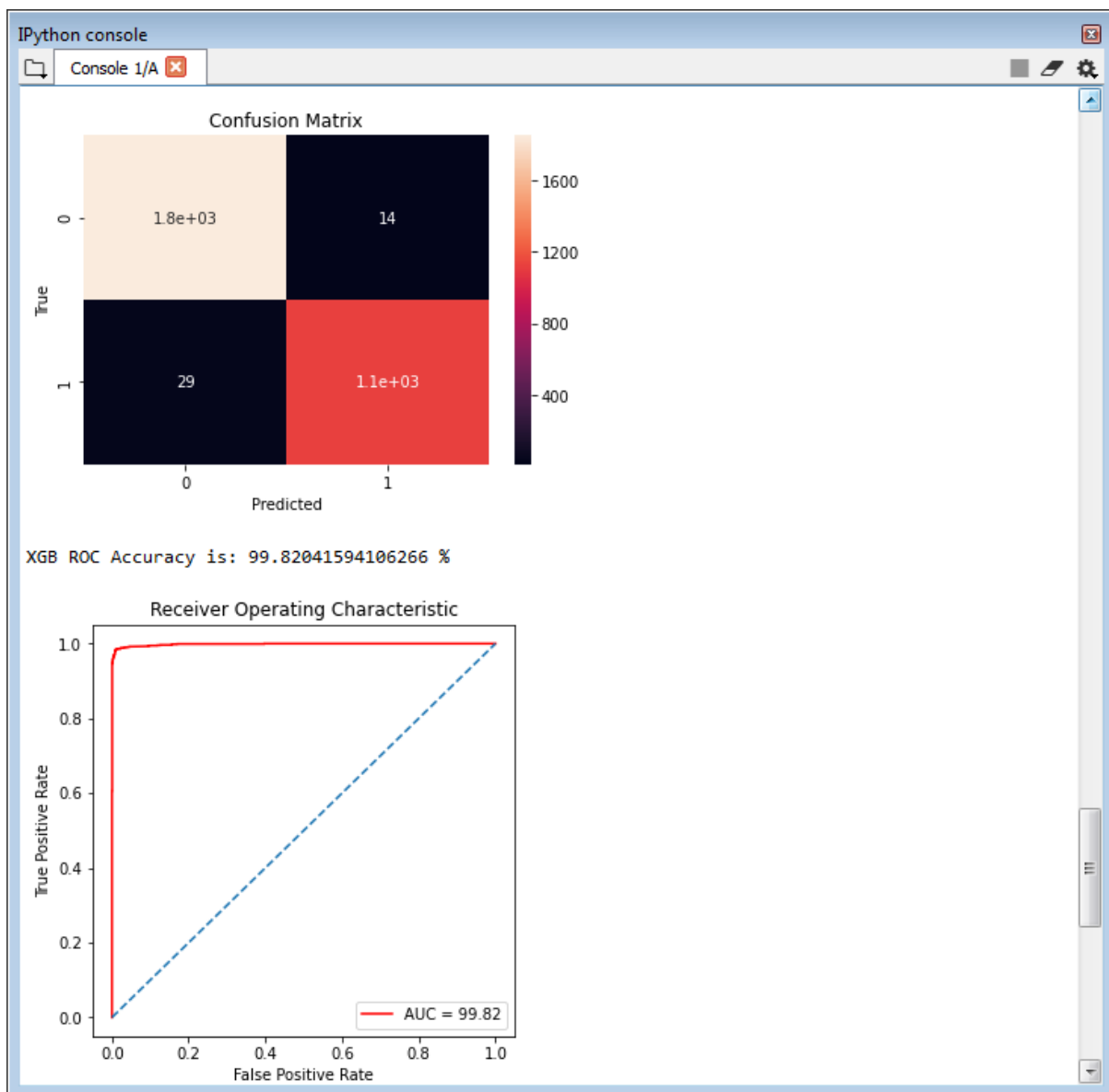


Figure.6.10 XGB ROC Accuracy

It's essential to note that the use of XGBoost in Android malware detection is not a one-size-fits-all solution, and its performance may vary based on the dataset and the specific problem being tackled. Hence, it is crucial to evaluate the model thoroughly and consider other algorithms like Random Forest and SVM for comparison.

## **CHAPTER 7**

### **CONCLUSION AND FUTURE ENHANCEMENT**

Classifiers based on ML are used in this work to predict final outcome. The Information from Android virus is inputted and processed beforehand methods are applied to clean the dataset and perform label encoding. The dataset is then divided into training and testing datasets which leads to feature selection methods are applied. PCA algorithm is used to perform feature reduction. Finally, a machine learning classification algorithm is employed in Android to forecast malicious software, and the results are evaluated on precision, roc precision and many more.

As a future work, there is potential to enhance the intended goals of the proposed classification and clustering techniques better accomplishment. Additionally, we aim to improve the collective layering structure as well as explore its applicability to further practical applications scenarios. In the future, there are several potential enhancements that could be made to improve the accuracy and effectiveness of Android malware detection. One possible enhancement is to incorporate more advanced machine learning techniques, such as deep learning and neural networks, into the detection process. Another potential enhancement is to incorporate additional features and data sources, such as information about network traffic and user behavior, into the detection models. Additionally, research could be done into developing more effective methods for identifying and removing obfuscated or hidden code within Android apps, which would improve the overall accuracy of detection. Finally, efforts could be made to improve the speed and efficiency of Android malware detection algorithms, allowing for faster and more reliable detection and removal of malicious apps.

## REFERENCES

- [1] Y. Mirsky, A. Shabtai, L. Rokach, B. Shapira, and Y. Elovici, "SherLock vs Moriarty: A Smartphone Dataset for Cybersecurity Research."
- [2] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention," *IEEE Transactions on Dependable & Secure Computing*, vol. PP, no. 99, pp. 1-1, 2018.
- [3] M. Xu, C. Song, Y. Ji, M. W. Shih, K. Lu, C. Zheng, R. Duan, Y. Jang, B. Lee, and C. Qian, "Toward engineering a secure android ecosystem: A survey of existing techniques," *Acm Computing Surveys*, vol. 49, no. 2, pp. 38, 2016.
- [4] T. Lei, Z. Qin, Z. Wang, Q. Li, and D. Ye, "EveDroid: Event-Aware Android Malware Detection Against Model Degrading for IoT Devices," *IEEE Internet of Things Journal*, 2019.
- [5] G. Tao, Z. Zheng, Z. Guo, and M. R. Lyu, "MalPat: Mining Patterns of Malicious and Benign Android Apps via Permission-Related APIs," *IEEE Transactions on Reliability*, vol. 67, no. 1, pp. 355-369, 2018.
- [6] Y. Sun, H. Song, A. J. Jara and R. Bie, "Internet of Things and Big Data Analytics for Smart and Connected Communities," in *IEEE Access*, vol. 4, pp. 766-773, 2016, doi: 10.1109/ACCESS.2016.2529723.
- [7] S. Sen, E. Aydogan, and A. I. Aysan, "Coevolution of Mobile Malware and Anti-Malware," *IEEE Transactions on Information Forensics & Security*, vol. 13, no. 10, pp. 2563-2574, 2018.
- [8] X. Ke, Y. Li, and R. Deng, "ICCDetector: ICC-Based Malware Detection on Android," *IEEE Transactions on Information Forensics & Security*, vol. 11, no. 6, pp. 1252- 1264, 2017.
- [9] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song and H. Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," in *IEEE*



Access, vol. 6, pp. 4321-4339, 2018, doi: 10.1109/ACCESS.2018.2792941.

[10] V. Avdiienko, K. Kuznetsov, A. Gorla, A. Zeller, and E. Bodden, "Mining Apps for Abnormal Usage of Sensitive Data." in ACM IEEE International Conference on Software Engineering, 2015.

[11] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. L. Traon, D. Ocateau, and P. McDaniel, "FlowDroid: Precise Context, Flow, Field, Objectsensitive and Lifecycle-aware Taint Analysis for Android Apps," *Acm Sigplan Notices*, vol. 49, no. 6, pp. 259-269, 2014..

[12] Y. Xue, G. Meng, L. Yang, H. T. Tian, and Z. Jie, "Auditing Anti-Malware Tools by Evolving Android Malware and Dynamic Loading Technique," *IEEE Transactions on Information Forensics & Security*, vol. 12, no. 7, pp. 1529-1544, 2017.

[13] T. Shibahara, T. Yagi, M. Akiyama, D. Chiba, and T. Yada, "Efficient Dynamic Malware Analysis Based on Network Behavior Using Deep Learning." in *IEEE Global Communications Conference*, 2016.

[14] Z. Lv, H. Song, P. Basanta-Val, A. Steed and M. Jo, "NextGeneration Big Data Analytics: State of the Art, Challenges, and Future Research Topics," in *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1891-1899, Aug. 2017, doi: 10.1109/TII.2017.2650204.

[15] H. Song, G. A. Fink, and S. Jeschke, *Security and Privacy in Cyber-Physical Systems: Foundations, Principles and Applications*. ISBN: 978-1-119-22604-8, Chichester, UK: Wiley-IEEE Press, 2017, pp. 1-472.

[16] E. B. Karbab, M. Debbabi, S. Alrabae, and D. Mouheb, "DySign: dynamic fingerprinting for the automatic detection of android malware." in *IEEE 11th International Conference on Malicious and Unwanted Software (MALWARE)*, 2016.

[17] Z. Lin, W. Rui, X. Jia, S. Zhang, and C. Wu, "Classifying Android Malware with Dynamic Behavior Dependency Graphs." in *IEEE Trustcom/bigdatase/ispa*, 2017.

[18] L. Jin, L. Sun, Q. Yan, Z. Li, and H. Ye, "Android Malware Detection," *IEEE Transactions*

on Industrial Informatics, vol. PP, no. 99, pp. 1-1, 2018.

[19] A. Pinandito, R. S. Perdana, M. C. Saputra, and H. M. AzZahra, "Spam detection framework for Android Twitter application using Naïve Bayes and K-Nearest Neighbor classifiers." in ACM international Conference on Software & Computer Applications, 2017.

[20] J. Zhu, Y. Song, D. Jiang and H. Song, "Multi-Armed Bandit Channel Access Scheme With Cognitive Radio Technology in Wireless Sensor Networks for the Internet of Things," in IEEE Access, vol. 4, pp. 4609-4617, 2016, doi: 10.1109/ACCESS.2016.2600633.

# APPENDIX

## CODE

### CODE SNIPPETS

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, auc,
roc_curve
import scikitplot as skplt
import seaborn as sns

#=====Input
Data=====
data = pd.read_csv("dataset/drebin-215-dataset-5560malware-9476-benign.csv")
print(data.head())
print(data.info())
print(data.describe())

#=====Pre-
processing=====
print("Total missing values : ",sum(list(data.isna().sum()))))

classes,count = np.unique(data['class'],return_counts=True)
#Perform Label Encoding
lbl_enc = LabelEncoder()
print(lbl_enc.fit_transform(classes),classes)
data = data.replace(classes,lbl_enc.fit_transform(classes))

```

#Dataset contains special characters like '?' and 'S'. Set them to NaN and use dropna() to remove them

```
data=data.replace(['?',S'],np.NaN,regex=True)
print("Total missing values : ",sum(list(data.isna().sum())))
data.dropna(inplace=True)
for c in data.columns:
    data[c] = pd.to_numeric(data[c])
data.head()
```

```
print("Total Features : ",len(data.columns)-1)
```

#benign and malignant count plot

```
plt.bar(classes,count)
plt.title("Class balance")
plt.xlabel("Classes")
plt.ylabel("Count")
plt.show()
```

#Dataset heatmap

```
corr= data.corr()
sns.heatmap(corr)
plt.title('Dataset Correlation Plot')
plt.show()
```

#=====model

selection=====

```
x = data.iloc[:, :-1].values
```

```
y = data.iloc[:, -1].values
```

```
train_x,test_x,train_y,test_y = train_test_split(x,y,test_size = 0.2, shuffle=True)
```

```
print("Train features size : ",len(train_x))
```

```
print("Train labels size : ",len(train_y))
```

```
print("Test features size : ",len(test_x))
```

```
print("Test labels size : ",len(test_y))
```

```

# Scaling the data to make it suitable for the auto-encoder
X_scaled = MinMaxScaler().fit_transform(x)
df1 = pd.DataFrame(X_scaled)

#PCA
pca = PCA()
x_pca = pca.fit_transform(x)
x_pca = pd.DataFrame(x_pca)
x_pca.head()

#=====Classification=====
=====

#pca = PCA(n_components=2)
pca_dims = PCA()
pca_dims.fit(train_x)
cumsum = np.cumsum(pca_dims.explained_variance_ratio_)
d = np.argmax(cumsum >= 0.95) + 1

pca = PCA(n_components=d)
X_reduced = pca.fit_transform(train_x)
X_recovered = pca.inverse_transform(X_reduced)
print("reduced shape: " + str(X_reduced.shape))
print("recovered shape: " + str(X_recovered.shape))

"RandomForestClassifier"

print('RandomForestClassifier')
print()
classifier = RandomForestClassifier()
classifier.fit(X_reduced,train_y)
X_test_reduced = pca.transform(test_x)

y_hat_reduced = classifier.predict(X_test_reduced)
rf_prob = classifier.predict_proba(X_test_reduced)[:,-1]
print("RF accuracy is: " + str(accuracy_score(test_y, y_hat_reduced))*100)

```

```

print()
print('Classification Report')
cr=classification_report(test_y, y_hat_reduced)
print(cr)
rf_cm = confusion_matrix(test_y, y_hat_reduced)
print(rf_cm)
print()
rf_tn = rf_cm[0][0]
rf_fp = rf_cm[0][1]
rf_fn = rf_cm[1][0]
rf_tp = rf_cm[1][1]
Total_TP_FP=rf_cm[0][0]+rf_cm[0][1]
Total_FN_TN=rf_cm[1][0]+rf_cm[1][1]
specificity = rf_tn / (rf_tn+rf_fp)
RF_specificity=format(specificity,'.3f')

print('RF_specificity:',RF_specificity)
print()

rf_mcc=((rf_tp*rf_tn)-
(rf_fp*rf_fn))/(np.sqrt((rf_tp+rf_fn)*(rf_tn+rf_fp)*(rf_tp+rf_fp)*(rf_tn+rf_fn)))
print('RF_Matthews Correlation Coefficient:',rf_mcc)
plt.figure()

skplt.estimators.plot_learning_curve(RandomForestClassifier() , train_x, train_y,
                                     cv=7, shuffle=True, scoring="accuracy",
                                     n_jobs=-1, figsize=(6,4), title_fontsize="large", text_fontsize="large",
                                     title="RF Digits Classification Learning Curve");

plt.figure()
sns.heatmap(confusion_matrix(test_y,y_hat_reduced),annot = True)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()

print()
rf_fp, rf_tp, thresholds = roc_curve(test_y, rf_prob)

```

```

rf_roc_auc = auc(rf_fp, rf_tp)*100
print('RF ROC Accuracy is:',rf_roc_auc,'%')
#LR ROC plot
plt.figure(figsize=(5,5))
plt.title('Receiver Operating Characteristic')
plt.plot(rf_fp,rf_tp, color='red',label = 'AUC = %0.2f' % rf_roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],linestyle='--')
plt.axis('tight')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()

""XGBClassifier""
print()
print('XGBClassifier')
print()
from xgboost import XGBClassifier

clf = XGBClassifier()
clf.fit(X_reduced,train_y)

yhat_reduced = clf.predict(X_test_reduced)
mlp_prob = clf.predict_proba(X_test_reduced)[:,1]
print("XGB accuracy is: " + str(accuracy_score(test_y, yhat_reduced))*100)
print()
print('Classification Report')
xgb_cr=classification_report(test_y, yhat_reduced)
print(xgb_cr)
xgb_cm = confusion_matrix(test_y, yhat_reduced)
print(xgb_cm)
print()
xgb_tn = xgb_cm[0][0]
xgb_fp = xgb_cm[0][1]
xgb_fn = xgb_cm[1][0]
xgb_tp = xgb_cm[1][1]

```

```

Total_TP_FP=xgb_cm[0][0]+xgb_cm[0][1]
Total_FN_TN=xgb_cm[1][0]+xgb_cm[1][1]
specificity = xgb_tn / (xgb_tn+xgb_fp)
xgb_specificity=format(specificity,'.3f')

print('XGB_specificity:',xgb_specificity)
print()

xgb_mcc=((xgb_tp*xgb_tn)-
(xgb_fp*xgb_fn))/(np.sqrt((xgb_tp+xgb_fn)*(xgb_tn+xgb_fp)*(xgb_tp+xgb_fp)*(xgb_tn+xg
b_fn)))
print('XGB_Matthews Correlation Coefficient:',xgb_mcc)
plt.figure()

skplt.estimators.plot_learning_curve(XGBClassifier() , train_x, train_y,
                                     cv=7, shuffle=True, scoring="accuracy",
                                     n_jobs=-1, figsize=(6,4), title_fontsize="large", text_fontsize="large",
                                     title="XGB Digits Classification Learning Curve");

plt.figure()
sns.heatmap(confusion_matrix(test_y,yhat_reduced),annot = True)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()

print()
xgb_fp, xgb_tp, thresholds = roc_curve(test_y, mlp_prob)
xgb_roc_auc = auc(xgb_fp, xgb_tp)*100
print('XGB ROC Accuracy is:',xgb_roc_auc,'%')
#LR ROC plot
plt.figure(figsize=(5,5))
plt.title('Receiver Operating Characteristic')
plt.plot(xgb_fp,xgb_tp, color='red',label = 'AUC = %0.2f % xgb_roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],linestyle='--')
plt.axis('tight')
plt.ylabel("True Positive Rate")

```



```
plt.xlabel('False Positive Rate')  
plt.show()
```

# PLAGIARISM REPORT

Plag Report Major NWC84125

## ORIGINALITY REPORT

<b>3</b> %	<b>3</b> %	<b>3</b> %	<b>1</b> %
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

## PRIMARY SOURCES

<b>1</b>	Huijuan Zhu, Yang Li, Ruidong Li, Jianqiang Li, Zhu-Hong You, Houbing Song. "SEDMDroid: An enhanced stacking ensemble of deep learning framework for Android malware detection", IEEE Transactions on Network Science and Engineering, 2020 Publication	<b>1</b> %
<b>2</b>	<a href="http://pure.royalholloway.ac.uk">pure.royalholloway.ac.uk</a> Internet Source	<b>1</b> %
<b>3</b>	<a href="http://api.openalex.org">api.openalex.org</a> Internet Source	<b>&lt;1</b> %
<b>4</b>	<a href="http://bip.imis.athena-innovation.gr">bip.imis.athena-innovation.gr</a> Internet Source	<b>&lt;1</b> %
<b>5</b>	<a href="http://dblp.dagstuhl.de">dblp.dagstuhl.de</a> Internet Source	<b>&lt;1</b> %

Exclude quotes On  
Exclude bibliography On

Exclude matches Off


<b>SRM INSTITUTE OF SCIENCE AND TECHNOLOGY</b> (Deemed to be University u/s 3 of UGC Act, 1956)		
<b>Office of Controller of Examinations</b>		
REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES <b>(To be attached in the dissertation/ project report)</b>		
1	Name of the Candidate (IN BLOCK LETTERS)	Arya Patel Ashish Kondragunta
2	Address of the Candidate	41 Sagar Bungalows, Gulab Tower Road, Sola Road, Thaltej, Pincode:380061. <b>Mobile Number:</b> 9978696733
3	Registration Number	RA1911031010084 RA1911031010125
4	Date of Birth	15/07/2001 12/12/2001
5	Department	Computer Science Engineering Specialization in Information Technology.
6	Faculty	Engineering and Technology, School of Computing
7	Title of the Dissertation/Project	Job Profile Upskilling Through Content Based Recommendation System Using Hadoop MapReduce
8	Whether the above project /dissertation is done by	<p><del>Individual</del> or group : (Strike whichever is not applicable )</p> <p>a) If the project/ dissertation is done in group, then how many students together completed the project : 2 (Two)</p> <p>b) Mention the Name &amp; Register number of other candidates :</p> <p>Arya Patel, RA1911031010084</p> <p>Ashish Kondragunta RA1911031010125</p>
9	Name and address of the Supervisor / Guide	Dr. K. A. Varun Kumar Assistant Professor Department of Networking and Communications SRM Institute of Science and Technology Kattankulatur - 603 203. <b>Mail ID:</b> <a href="mailto:varunk@srmist.edu.in">varunk@srmist.edu.in</a> <b>Mobile Number:</b> 9787826259

10	Name and address of Co-Supervisor / Co- Guide (if any)	NIL  <b>Mail ID: Mobile Number:</b>
----	---	---

11	Software Used	Turnitin		
12	Date of Verification	09 – May -2023		
13	<b>Plagiarism Details: (to attach the final report from the software)</b>			
Chapter	Title of the Chapter	Percentage of similarity index (including self citation)	Percentage of similarity index (Excluding self citation)	% of plagiarism after excluding Quotes, Bibliography, etc.,
1	INTRODUCTION	1%	1%	1%
2	LITERATURE SURVEY	1%	1%	1%
3	TECHNICAL SPECIFICATIONS	0%	0%	0%
4	ARCHITECTURE	0%	0%	0%
5	MODULES	0%	0%	0%
6	PROJECT DEMONSTRATION AND RESULTS	0%	0%	0%
7	CONCLUSION	0%	0%	0%
8	FUTURE ENHANCEMENTS	1%	1%	1%
9				
10				
Appendices		3%	3%	3%
I / We declare that the above information have been verified and found true to the best of my / our knowledge.				
Signature of the Candidate		Name & Signature of the Staff (Who uses the plagiarism check software)		
Name & Signature of the Supervisor/ Guide		Name & Signature of the Co-Supervisor/Co-Guide		
Name & Signature of the HOD				

# PAPER PUBLICATION STATUShgy

WhatsApp 6354477117
editor@ijrar.org
ijrar.org



**INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS (IJRAR.ORG)** Impact Factor: 7.17  
Calculated by Google Scholar

International Peer Reviewed & Refereed Journals, Open Access Journal

ISSN Approved Journal No: E-ISSN 2348-1269, P- ISSN 2349-5138 | ESTD Year: 2014

Scholarly open access journals, Peer-reviewed, and Refereed Journals, AI-Powered Research Tool, Multidisciplinary, Monthly, Indexing in all major database & Metadata, Citation Generator, Digital Object Identifier(DOI)

**Your Paper Acceptance details as per below Complete Step 1 and Step 2 and publish within 1 to 2 days.**

**Dear Author, Congratulation!!**

Your manuscript with Registration ID: **IJRAR\_ 265075** has been accepted for publication in the International Journal of Research and Analytical Reviews (IJRAR) | [www.ijrar.org](http://www.ijrar.org) | International Peer Reviewed & Refereed Journals, Open Access Journal. IJRAR is Peer Review Journal, Refereed Journal, Peer Reviewed Journal Referred Journal and Indexed Journal Open Access Journal Online and Print Journal

UGC Approved Journal NO: 43602(19)  
IJRAR Impact Factor: 7.17 (Calculated by Google Scholar)

Check your paper status: <https://ijrar.org/track.php> or <http://ijrar.org/Authorhome/alogin.php>

Online Payment Link for Indian author: [http://ijrar.org/pay\\_form\\_2.php](http://ijrar.org/pay_form_2.php)  
Online Payment Link for Foreign/international author: [https://ijrar.org/pay\\_form\\_international.php](https://ijrar.org/pay_form_international.php)

**Your Paper Review Report :**

Registration ID: IJRAR – 265075	
Title of the Paper:	SEMDROID: AN ENHANCED STACKING ENSEMBLE? FRAMEWORK FOR ANDROID MALWARE DETECTION.
Accepted or Not:	Accepted
Criteria	Points out of 100%
Continuity	87%
Text structure	90%
References	91%
Understanding and Illustrations	95%
Explanatory power	94%
Detailing	90%
Relevance and practical advice	94%

**Track Your Paper Details:**

Paper Track Link 1: <https://ijrar.org/track.php>  
Paper Track Link 2: <http://ijrar.org/Authorhome/alogin.php>

**Online Payment link Details:**

Indian author: [http://ijrar.org/pay\\_form\\_2.php](http://ijrar.org/pay_form_2.php)  
Foreign/international author: [https://ijrar.org/pay\\_form\\_international.php](https://ijrar.org/pay_form_international.php)

Overall Assessment (Comments.) Paper Accepted: YES  
Unique Contents: 96% Paper Accepted Complete Payment