Android Malware Detection and Family Classification

Author

Shoaib Hussain

00000205447

Supervisor

Dr. Farhan Riaz

DEPARTMENT OF COMPUTER ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

September 2021

Android Malware detection and Family Classification

Author
Shoaib Hussain
00000205447

A thesis submitted in partial fulfillment of the requirements for the degree of
MS Computer Engineering

Thesis Supervisor
Dr. Farhan Riaz

Thesis Supervisor's Signature: _____

DEPARTMENT OF COMPUTER ENGINEERING
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,
ISLAMABAD
September 2021

Declaration

I certify that “Android Malware Detection and Family Classification” is my own work. The thesis has not been submitted elsewhere for review. The use of content from other sources has been properly recognized and referred to.

Signature of Student

Shoaib Hussain

00000205447

Language Correctness Certificate

The thesis has been proofread and is free of most textual, syntax, semantic, grammatical, and spelling errors. The format of the thesis is also dictated by the university.

Signature of Student

Shoaib Hussain

00000205447

Signature of Supervisor

Dr. Farhan Riaz

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached.

Signature of Student

Shoaib Hussain

00000205447

Signature of Supervisor

Dr. Farhan Riaz

Copyright Statement

Acknowledgements

By the grace of the lord above, my Master’s thesis is finally complete. It was quite a burdensome exercise that couldn't have been finished without the help of God Almighty and the strength that he gave to me.

I'm additionally thankful to all those, particularly my loved ones, who assisted me with the thesis at all times. Their support was what made me accomplish this task. Without them, it would not have been conceivable.

Lastly, I would like to offer my sincere thanks to my incredible supervisor and the entire committee: Dr. Farhan Riaz, Dr. Farhan Hussain, Dr. Ali Hassan and Dr. Ahsan Shehzad. Without their support – in every sense of the word – and constant guidance, this thesis would never have been conceived or completed. I cannot thank them enough for their role in the completion of this thesis and report.

This thesis is dedicated to my exceptional parents

For their everlasting love, encouragement, and support

Abstract

Android has become a predominant mobile operating system lately. Google with the help of Open Handset Alliance setting out to create open standards for smart phones have prompted a gigantic development in the digital world. Given the growth and development of smartphone devices and their related application stores, Malware detection is a developing issue. Volume of new applications is excessively enormous to physically analyze every application for malicious activity. Keeping this in view this research presents a method to detect android malware and further classify it to four malware categories and thirty nine malware families. The classification model has been built around reduction of redundant features and employing three machine learning algorithms (Random Forest, KNN and SVM algorithms) in binary classification and Random forest algorithm for category and family classification. The proposed methodology performs reasonably well for most of the classes achieving around an accuracy of 95% on binary classification. Proposed method provides the accuracy of 84% on malware category classification and accuracy of 66% for Malware family classification.

Key Words *Android malware, Random Forest, KNN, SVM, Feature reduction, Machine Learning, Malware category, Malware family*

Table of Contents

List of Figure

List of Table

LIST OF ABBREVATIONS

API	Application program interface
APK	Android application Kit
DT	Decision Tree
RF	Random Forest
KNN	K-Nearest Neighbor
ML	Machine Learning
OEM	Original equipment manufacturer
PCA	Principal Component Analysis
PIN	Personal identification number
RT	Random Tree
SVM	Support vector machine
SDK	Software Development Kit
TWRP	Team Win Recovery Project

CHAPTER 1: INTRODUCTION

Android has become a predominant mobile operating system lately. Google with the help of Open Handset Alliance setting out to create open standards for smart phones have prompted a gigantic development in the digital world. Smart phones have picked up admiration in the presence of different types of applications which provide services like online shopping, business and educational apps, traveling and location based apps, online gaming, banking and other utilities other than conventional services like calls and Texting.

In 2012, there were 1.06 billion smartphone users, which grew to 2.5 billion by 2016, and 3.6 billion by 2020. In 2023, smart phone users are predicted to reach 4.6 billion [1].

The ongoing growth of Google's android OS's multiple versions, with each new one adding greater features, quicker internet access, and better video and audio, is one of the reasons for its success. Android 1.0, the initial commercial version, was released in September 2008, followed by Android 1.1 in February 2009 [2].

1.1 Recent Statistics on Android Malware Threats

In 2010, android devices accounted for only around 20% of global smartphone sales [3]. The Android operating system's development and open-source nature have aided its rapid growth. A report by Gartner Statistics [4] shows rapid growth in smart phones sales and android occupies most of the share in it with 82%. In year 2020 out of 3.6 billion smartphones users' android occupies 74 % of users [5].

Rapid growth in development of android operating system has exposed it to malware attacks. Forbes Report in 2014 [6] showed that 97% of malware attacks in 2013 were targeted towards android. Android cell phones are immensely vulnerable to malware outspread as it permits applications to use the resources when the user allows permissions purposely or unconsciously.

Figure 1.1 shows in the year 2015, 8.2 billion malware attacks were recorded. In 2017 total number of malware attacks were 8.6 billion and increased by 22 percent in a year to reach 10.5 billion in 2018. In 2019 malware attacks were reduced by almost 6 percent (9.9 billion) [7].

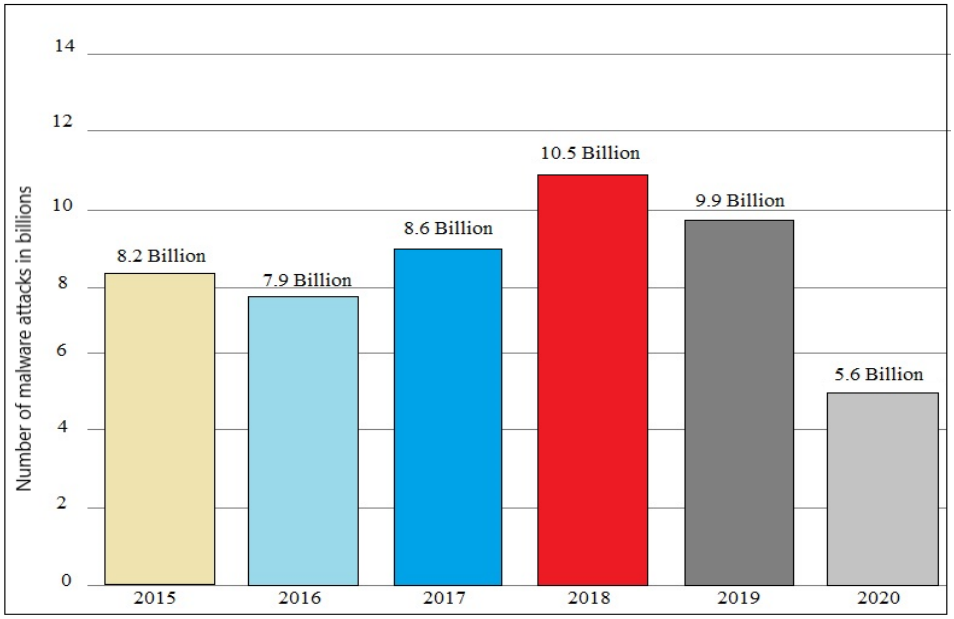

Figure 1. 1 Malware attacks statistics from 2015 to 2020 [7]

Data privacy and security is the biggest challenge these days, ransomware is a type of malware that encrypts files. Attacker then asks money from user in exchange for data access. In 2018, ransomware attacks grew by 350 percent around the world.

Ransomware attacks are predicted to cost $6 trillion per year by 2021. 75 percent of ransomware-infected businesses had up to date end-point security. Each year, ransomware costs companies around $75 billion. FedEx lost $300 million in the first quarter of 2017 due to the NotPetya ransomware outbreak. After being attacked by the ransomware attack in March 2018, Atlanta, Georgia has invested more than $5 million to repair its computer network [8].

According to Forbes [9], ransomware was the most widespread type of malware in 2016. Ransomware was responsible for 18 percent of all malware payloads distributed through spam and exploit kits in first month of 2016. After 10 months with an increase of 267%, it had bloomed to represent 66% of malware payloads.

Figure 1.2 shows in 2016, 638 million ransomware attacks were recorded. In year 2018, total number of ransomware attacks were 204.24 million. According to a global cyber security yearly report, 304 million ransomware attacks were registered in 2020, up 62% from the previous year (187.9 million in year 2019) [10].

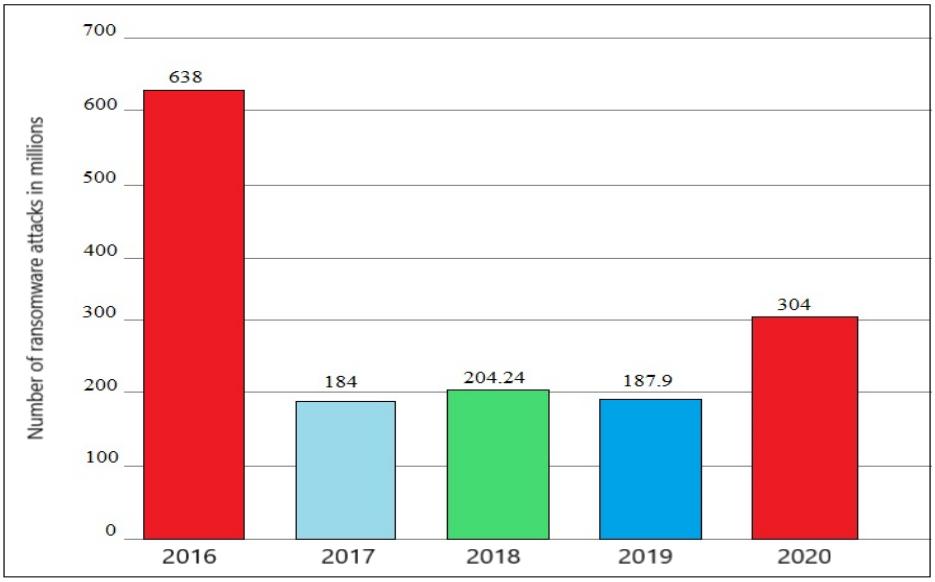

Figure 1. 2 Number of ransomware attacks from 2016 to 2020 [10]

Thailand experienced the highest percentage of users affected by ransomware (9.57%) from 2017 to 2018. UAE and Iran are second and third with 8.67 percent and 8.47 percent respectively [11].

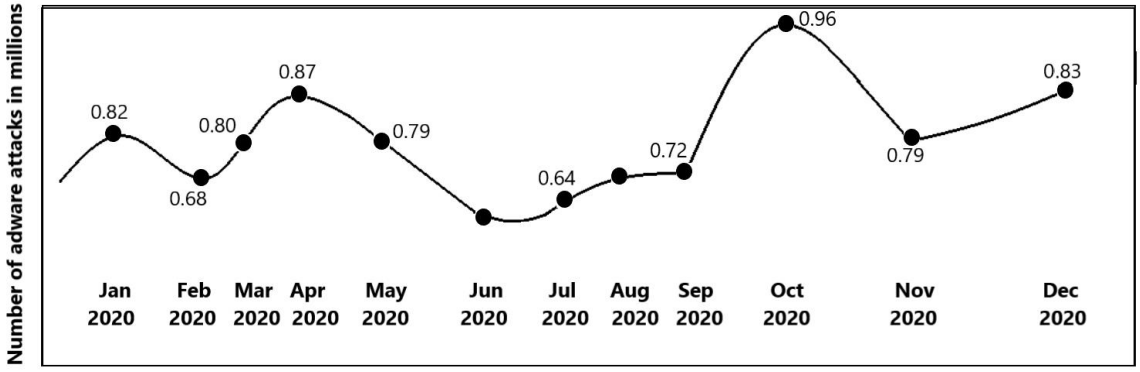

Figure 1. 3 Adware attacks per month in year 2020 [12]

Adware shared the 14.62 % of total android malware attacks in 2020. Only in month of January, 820000 adware attacks were recorded which increased to 870000 in April and reached the peak in October with 960000 adware attacks [12]. Figure 1.3 shows the adware attacks per month in year 2020.

1.2 Threats

Mobile malware threats are part of a complex ecosystem, but some of the most frequent mobile Malware are often used to deliver highly targeted attacks. For instance, DroidDream, the fifth-most-common mobile virus, generates a unique identity for the phone and waits for more instructions from its operator, all while working in the background even without user's permission. Another case [13] is AndroidExploit Masterkey, the third-most-common mobile malware, which alters APKs, which is used to install apps on Operating system. A hacker can essentially transform any genuine application into a malware and because of that consumer perception of security threats continues to fall behind the fact. Consumers who may never use unknown sources for installing apps on their device could unconsciously download malware from unverified sites/sources on their smart phones when they click on links in SMS. Therefore, SMS phishing campaigns aim to spread android malware are more successful, especially when it's customized to a particular company's client base. Customers are also known for being reluctant to upgrade their smartphones' operating systems. Therefore, it's no wonder that malware generally seen in user attacks, like the Basebridge Trojan, takes advantage of flaws in outdated mobile systems. Worse still, a sizable percentage of smartphone users actually attempt to root their smartphones to gain access to illegal application stores or to get free content.

1.2 Mobile Banking

In 2020, 1.9 billion people throughout the world utilized online banking services. By 2024, this number is anticipated to reach 2.5 billion [70]. Mobile payments have totaled $503 billion in 2020. Figure 1.4 shows the growth of active Online Banking Users Worldwide.

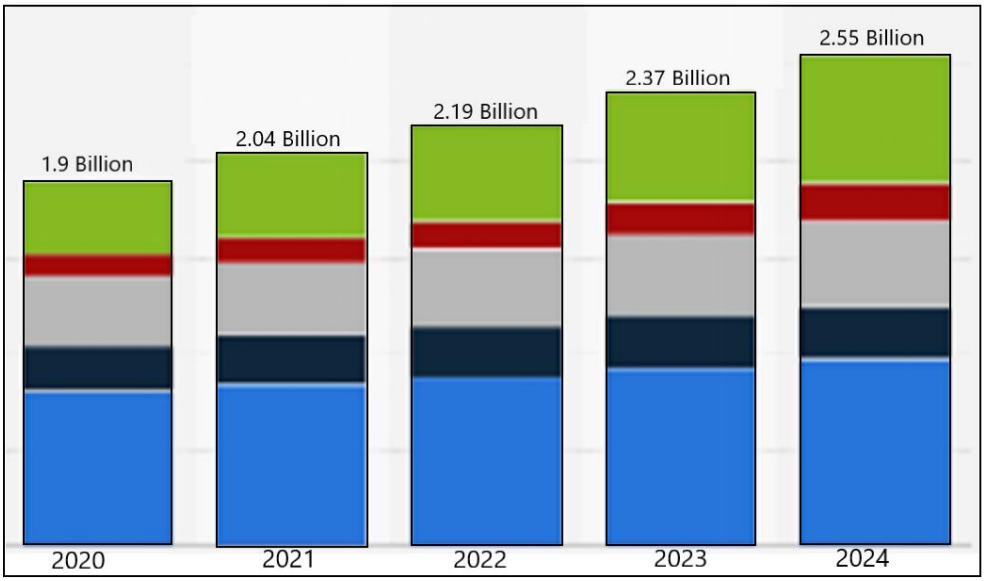

Figure 1. 4 Active Online Banking Users Worldwide [70]

Since 2015, the number of mobile application fraud transactions has risen by approximately 600%. According to US mobile banking statistics, 89 percent of US bank account holders utilize mobile banking to manage their accounts. In 2019, 14,392 breaches generated more than $40 million in mobile fraud damages.

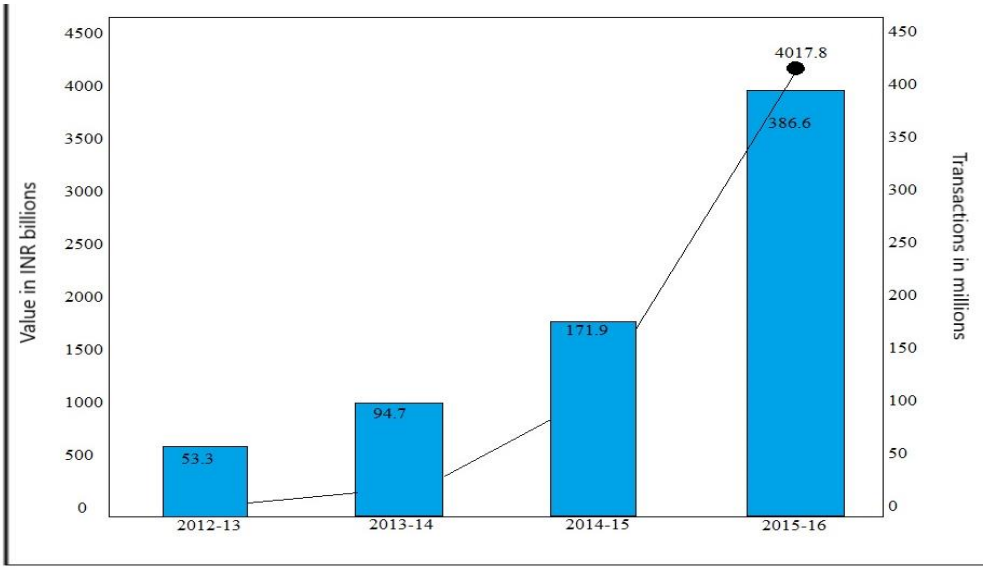

Figure 1. 5 Growth of Mobile Banking in India [71]

Mobile banking has grown in India from INR 53.3 billion in 2012-13 to reach INR 4017.8 billion in 2015-16 [71]. Figure 1.5 shows the graph of growth from year 2012 to 2016 in mobile banking in India. By the end of March 2018, there were 3.1 million mobile phone banking customers in Pakistan. In 2018, these customers used mobile phone banking applications to complete 5.9 million transactions valued PKR 112.8 billion. There has been an increased uptake of M-wallets within customer oriented mobile money transactions in Pakistan since 2015 [72]. Figure 1.5 shows the growth of mobile money transactions in Pakistan from year 2015 to 2020.

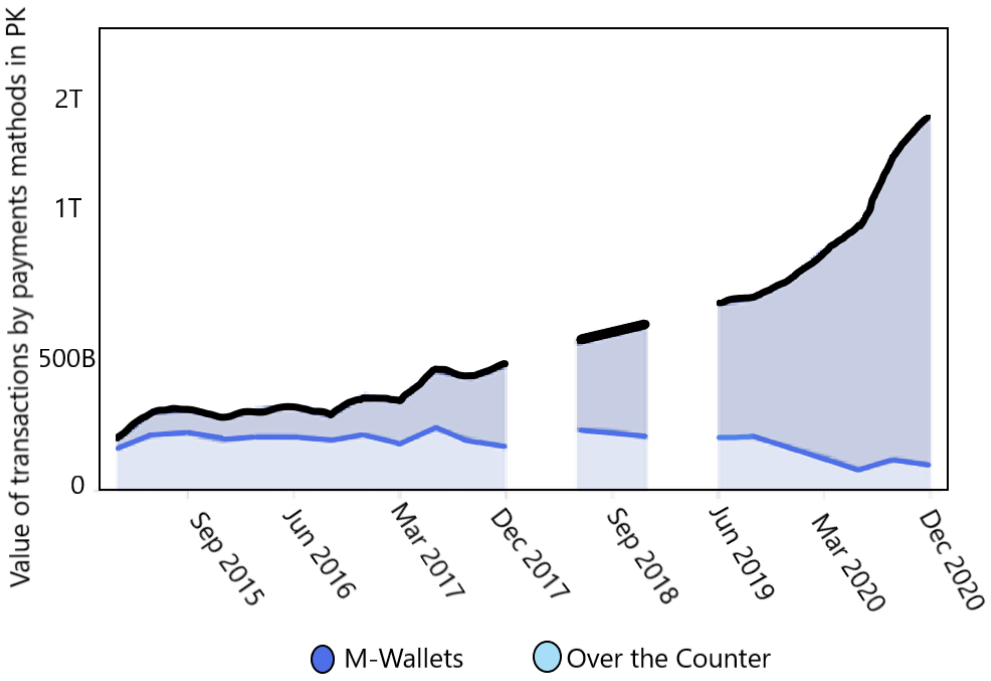

Figure 1. 6 Growth of Mobile Money Transactions in Pakistan [72]

1.4 Android Platform

The Android OS is depicted in Figure 1.4. Android is a Linux-based open source software. It creates an environment in which many applications can run at the same time. All of these apps are signed and placed in application sandboxes that are linked to their signatures. The apps sandboxes are the permissions available to the application. The rights open to the application are known as application sandboxes. The Android Runtime is a framework for designing apps

that interface with the operating system, providing system services, platform APIs, and message formats. Apps with an application sandbox are used to deploy system services. There really is no super user who has full access to smart phone above the kernel. The Dalvik virtual machine runs Android apps, which are written in Java. They are assisted by the app framework which provides a single interface for widely used features. A number of libraries exist that make it simple for apps to include graphics, secure and protected communication, and databases. The Bionic Standard Library is a libc for embedded devices based on BSD. For the Android updates, kernels from Linux 2.6 versions are cut down [13].

1.5 Motivation

Given the growth and development of smartphone devices and their related application stores, Malware detection is a developing issue. Volume of new applications is excessively enormous to physically analyze every application for malicious activity. With so many app stores including Google app store, malware applications have begun to spread. Therefore, Android attracts malicious attacks.

When user allows the permissions purposely or unconsciously, applications has access to android user's private and sensitive data. It is critical to be able to identify malware apps and the characteristics that they utilize to access sensitive information in order to keep Android users safe from malware attacks and threats.

1.6 Problem Statement

Our goal is to analyze and systematically classify the android malware into categories and families. Android security depends on a permission based system that limits access to critical resources for android apps. Before proceeding with the installation, the Android client should recognize the set of permissions that an application requires. This is to inform Android users about the application's resource needs as well as the risks associated with its installation and use. It has two issues. The first is that usually user don't know enough of existing risks and threats. The subsequent one is that Android doesn't show the particular permissions and resources an application required making it tougher to detect malware applications and laying the groundwork for additional malware attacks. Canadian Institute for cybersecurity has released the publicly available dataset CICInvesAndMal2019 [14] which comprises of intents and permissions as static features and API calls and generated log files as dynamic features.

Although second part of dataset has improved the results but still there is a room for improvement.

The purpose of this research is feature reduction and accurate prediction of this dataset into malware and benign apps and further classification of this dataset into categories and families of malware.

1.7 Aims and Objectives

The following are the research's main goals:

- Explore the dataset to be used for classification
- To reduce the dimensionality and features of dataset to be used for classification
- Develop an algorithm for classification of malware and benign apps
- Develop an algorithm for classification of the malware apps into categories and families of malware.

1.8 Structure of Thesis

The thesis is organized as follows:

Chapter 2 It consists of literature review as well as key work done by researchers in past few years for malware classification

Chapter 3 Includes discussion on machine learning algorithms and feature reduction techniques

Chapter 4 Provides the insight of dataset in detail with discussion about layers and features. It is also comprised of proposed methodology thoroughly.

Chapter 5 Provides all the experimental results along with relevant figures.

Chapter 6 Concludes the thesis and reveals future scope of this research.

CHAPTER 2: LITERATURE REVIEW

With the development and rapid growth of smartphone devices and their related application stores, Malware detection is a developing issue. Volume of new applications is excessively enormous to physically analyze every application for malicious activity. With so many app stores including Google app store, malware applications have started to grow. Therefore, Android attracts malicious attacks.

2.1 File Structure

2.1.1 AndroidManifest.xml

Manifest file of android application describes all the important information in APK file,

Important parts of APK file are:

1. Package Name: In both smart phone and the Play Store, a package name is a unique identifier for an application.
2. App Components: The behaviour of an application is defined by its services, activities, and capacity to interface with the OS, Content Providers, and Broadcast Receivers.
3. Manifest Permissions: It describes what an application requires to access important information of system or applications

2.1.2 Android SDK

SDK is a collection of tools that makes it possible to create APKs. Developers have access to a wide range of APIs that provide them access to numerous functionalities provided by local libraries and components.

APIs in general change after the release of each version of Android, hence make loop holes in framework. To recognize APIs between every update, Google marks them as per their level [15].

There are many datasets and several methods in literature for detection of malware and analysis by use of different machine learning techniques for classification and detection. This chapter will cover all those important researches in this field.

2.2 Literature Review

There are two primary ways to android malware detection: static and dynamic analysis. Static analysis, APKs are decompiled before installation, Static analysis can result in a quick and secure android malware detection. To identify malicious apps, researchers often employ data flow tracking as well as associated attribute information as well from APK files to identify malicious apps. Qin [16] examines risky permissions and saves them in a database, then extracts permissions from an unknown Android app and compares them to the harmful permissions to arrive at a judgement.

Several studies [17] [18], have looked into malicious static signature based on the analysis of Android permissions. Indeed, the permissions granted to a certain application can provide a basic indication of harmful behavior's possible consequences. It is possible to detect the presence of possible malware before it is executed by examining these permissions in combination with a semantic analysis of the application code. However, this method cannot be used over the phone and is wasteful in terms of resources.

Another technique for static malware detection is studying API calls from mobile applications. This technique is present in various works such as (Fan et al., 2015) [19], (Wu et al., 2012) [21] and (Zou et al., 2015) [20]. In the work of Fan et al. (2015) [19], API calls are being used to help detect malicious applications that have code embedded in normal applications. They therefore study API calls for non-malicious applications and train a model of normal behavior reflected by the lists of API calls.

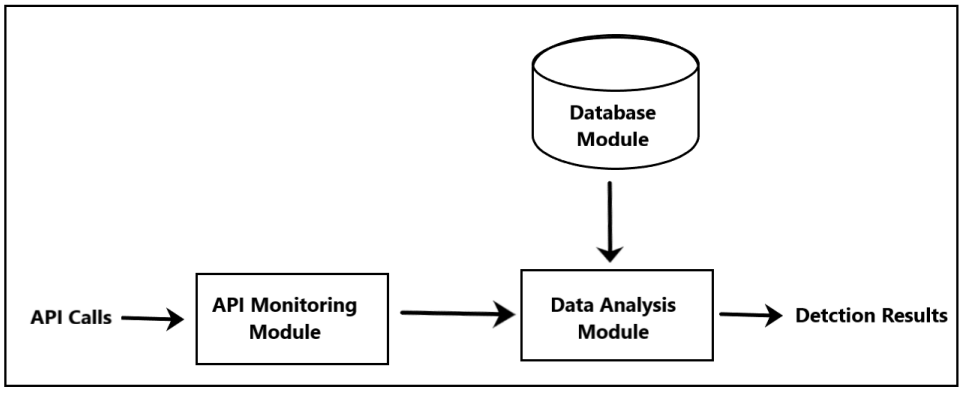

Figure 2. 1 Detection model used in Fan et al. (2015) [19]

Any deviation from this pattern is considered malicious behavior. An illustration of the detection model is shown in Figure 2.1. In (Zou et al., 2015), the authors propose a malware detection tool on Android systems called DroidMat. This tool is also based on the static analysis of mobile applications through the study of API calls. Indeed, the authors claim that these calls provide information about the types of operations that are likely to be performed by a given application. For example, some malicious applications tend to extract sensitive information from the smart phone like the serial number through the background call to the API. For normal applications, this call is not made in the background and is displayed on the phone screen

The Genome project is one of the earliest openly accessible endeavors in 2012 [22]. They presented the very first comprehensive collection of dataset containing android malware samples with 1260 samples of malware from year 2010 to year 2011 in 49 different malware families. In Genome project they utilize static analysis strategies to classify and characterize behavior of malware by assessing the installation, activation and data payload samples. The techniques were focused on studied permission, static test and on statically examined parts of malicious source code and on tracked API calls. They likewise utilized their suggested dataset on genuine smartphones to analyze the adequacy of the current antivirus software. For this dataset they checked the detection results with four mobile antivirus software including AVG, Trend Micro, Lookout and Norton. Out of 1260 samples AVG detected 689 Samples (54.7%), Lookout detected 1003 (79.6%) samples, Norton detected 254 samples (20.2%) and Trend Micro detected 966 (76.7%) samples.

Aung et al. [23] built up an ML based model to detect malware which screens different permission features and event to detect malware applications from benign ware applications. For clustering they used K-means algorithm. In Classification between malware applications and benign ware applications they used Random Forest (RF) and Decision Tree (DT) algorithms. The experiments on two distinctive datasets showed detection rate of 90% on average

In 2014 Drebin [24] dataset was introduced which gave 123,453 benign and 5560 malicious samples covering total of 20 families. They prepared their Classification framework on the static features include App components, hardware components, network addresses, requested

permissions and filtered intents to assess their dataset. Figure 2.2 shows the schematic representation of drebin methodology. Drebin methodology involves a comprehensive static analysis to obtain feature sets from various sources and analyze them in an expressive vector space.

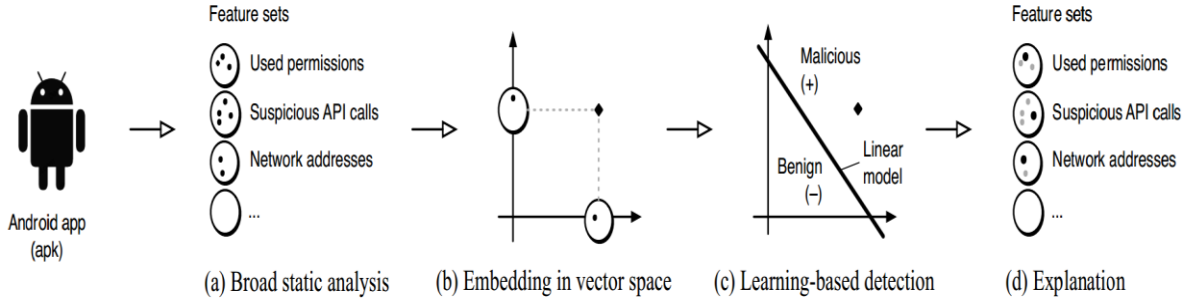

Figure 2. 2 Schematic representation of analysis steps performed by drebin [24]

This process involves

a. Broad static analysis. At first step, DREBIN examines an Android app statically and extracts various feature sets from manifest and dex code.
b. Embedding in vector space. The derived set of feature are then mapped to a joint vector space, which allows geometric analysis of patterns and combinations of features.
c. Learning-based detection. Use of Machine learning techniques to identify malware by using embedding of feature sets.
d. Explanation. The final step identifies features which contribute to the identification of a malware app and presents them to the user to describe the detection process.

In spite of the fact that DREBIN adequately recognize malicious data, it presents the inherent limitations of static analysis.

SAPIMMDS Dataset [25] from KISA (Korea Internet Security Agency) has 1776 benign samples and 906 samples of 13 malware families. In this dataset with the help of byte code memory dump techniques they traced the doubtful sequences of API calls to take out API call patterns for specific malicious functionality through this data.

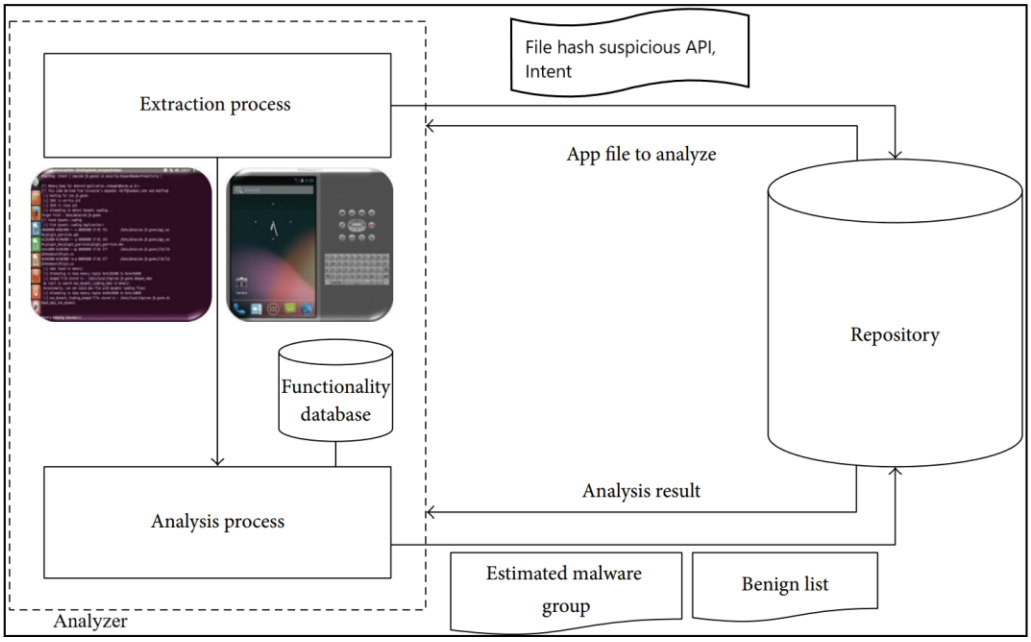

Figure 2. 3 System architecture for SAPIMMDS Dataset [25]

Figure 2.3 show an analyzer and a repository comprise the system. On the server, which runs Ubuntu 12.04 LTS (64-bit), the analyzer and repository are installed. Analyzer is built with an Android emulator running on Android 4.1.2 and the Python programming language. Confusion matrix for malware detection for SAPIMMDS Dataset shows the performance of the detection algorithm. 8 benign samples were identified as malware, while 147 malware samples were identified as benign, accounting for 16.23% of all malware samples. Precision recall and accuracy, in summary, are 0.990, 0.838, 0.942, and 0.907, respectively.

Table 2. 1 Confusion matrix for malware detection for SAPIMMDS dataset [25]

Category	**Actual class**	
	Malware Applications	**Benign Applications**
Estimated class		
Malware Applications	759	8
Benign Applications	147	1,768

Andro-Dumpsys (2016) [26] involves comparing malware creator centric and malware centric data for similarity. By exploiting malware samples' footprints that are basically equivalent to unique behavior characteristics, authors use Andro-Dumpsys to classify malicious samples into similar behavior groups. Test results reveal that Andro-Dumpsys is indeed scalable and capable of classifying malware families with less false positives and negatives.

Andro Dumpsys dataset incorporates 1776 benign samples and 906 malware samples. This dataset contains malware attributes with intent-based features to classify malicious data. For Andro Dumpsys dataset they have adopted the permission distributions, intents, API call sequences, use of commands for executing fabricate files and serial number of a certificates and as feature vectors.

On the basis of above mentioned features they led profiling systems as indicated by the connections among the op-code and byte-code for all malicious families. In the classification engines and the detection on server side, authors determined the similarity score between every client-side APK request and their profiling patterns.

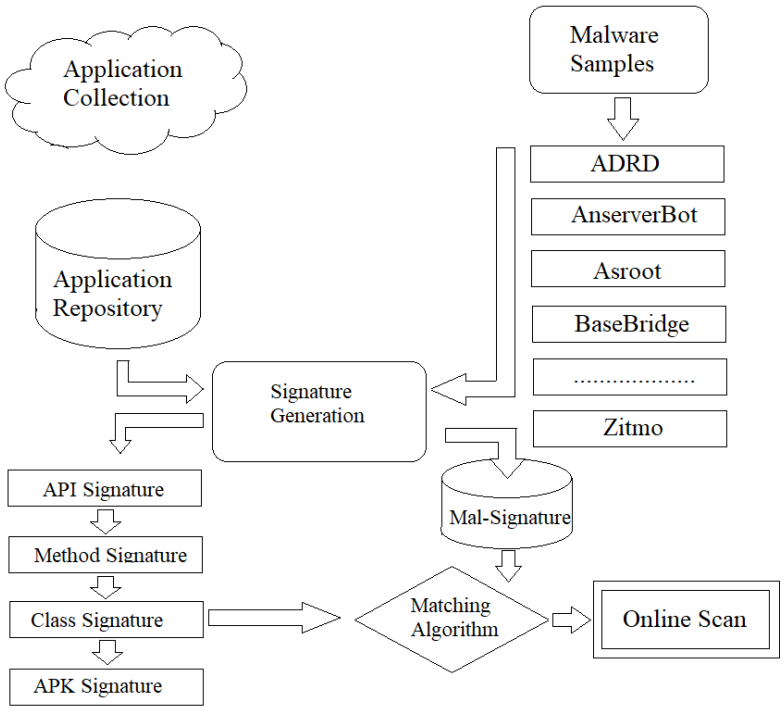

Figure 2. 4 Hierarchical signatures approach used by wang and wu (2015) [27]

A method of detecting malware on Android systems has been proposed by Wang and Wu (2015) [27]. In this work they used hierarchical signatures which combine both the signatures of API calls, the hash signatures of malicious applications, as well as those of the classes and methods of these applications. A description of this approach is shown in the figure 2.4. Combination of the two detection techniques (permission and API calls) is also exploited by several works [28], [29], [30], [31] and [32]. In this work, the robustness of the two detection tools is combined in order to improve the detecting system's overall performance. In order to classify the analyzed apps, this work use machine learning methods.

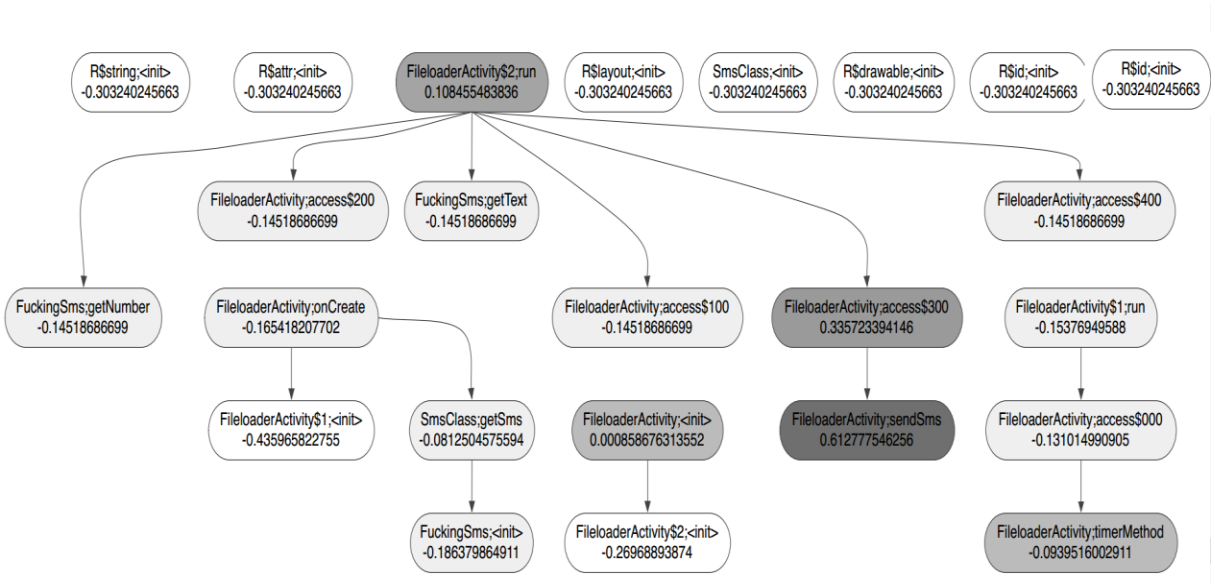

Figure 2. 5 Structure and logic of malicious code function calls [33]

The malicious behavior of malware on mobile phones can also be reflected in the static analysis of malicious code and the extraction of function call graphs. This graph illustrates the structure and logic of malicious code function calls and organizes them in the form of a graph. An example of function call graphs of the "Android: RuFraud-C" malware belonging to the malicious "FakeInstaller" family is illustrated in figure 2.5 taken from Gascon et al. (2013) [33]. Dark nodes indicate the call structures of malicious functions detected by this tool. As for the other nodes, they represent the normal call structures.

Behavioral signature detection is done at runtime of applications and looks at malicious behavior. One of the most widely used tools for behavioral detection of mobile malware is

interception and analysis of system calls between applications and the Android system kernel. In Isohara et al. (2011) [34], a study by behavior of Android applications through the analysis of log files containing the list of system calls made at the kernel level of the Android system. These files are analyzed with pattern matching algorithms to detect the presence of previously determined system call signatures. These signatures are in regular expression format whose construction has not been detailed.

In (Lin et al., 2013b) [35], a detailed study for the construction of behavioral signatures of system calls was carried out with the aim of detecting malware on Android systems. This technique consists of taking two types of applications: normal applications and malicious applications grouped by malware family. For each family, it extract the system call sequences and only keep the sequences that are common. A sequence filtering step must then be done by additionally considering the system call sequences of normal applications. According to Lin et al. (2013b), a detection accuracy of up to 95.97% could be obtained with this type of signatures. However the performance of detection always depends on the signature construction phase and the ability to find applications belonging to the same malware families.

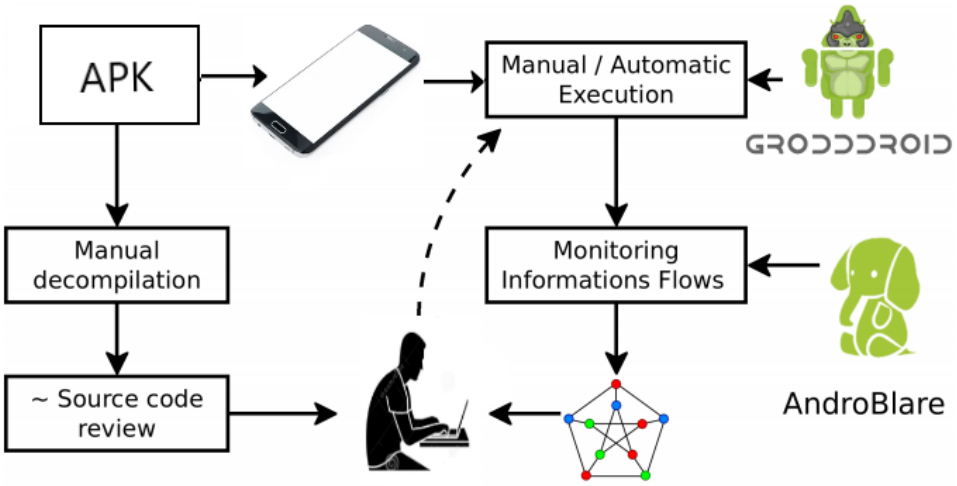

Figure 2. 6 Methodology for kharon dataset [36]

In Kharon dataset (2016) [36] they have created the dataset by running the malware samples on actual mobile phones. By reversing the code of each malware, the authors were able to dissect them. Figure 2.6 shows the methodology for Kharon dataset.

In order to obtain their precise behavior, malware was used in a controlled and supervised real mobile device. 7 malware samples were used and for tracking the flow of information between the system objects like files, sockets at system level they used AndroBlare tool. Mainly they have used two analysis methods: static analysis and dynamic analysis. The authors also used graph models of flow of information produced by an execution to summarize their behavior. In light of the findings, it was also discussed that the majority of malware includes triggering methods which delay and disguise their malware behavior

Lashkari et al. 2017 [37], introduced the AAGM dataset which has 1900 samples from 2008 to 2016 out of which 400 are malware and 1500 are benign samples. Dataset contains twelve families of benign, general malware and adware categories. Samples were introduced on genuine cell phones and started performing usual user interaction cases for capturing network traffic. Machine learning algorithms were used for analysis.

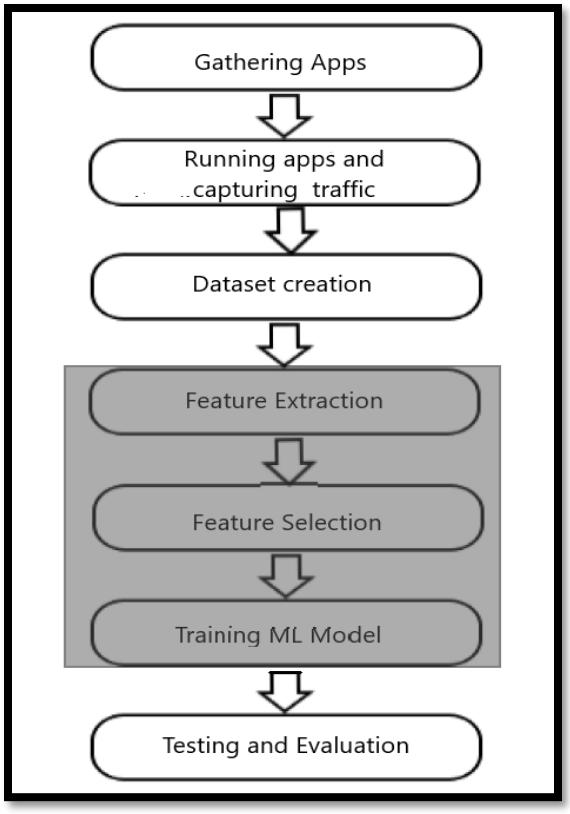

Figure 2. 7 Steps for acquiring AAGM dataset [37]

Figure 2.7 shows the steps for acquiring the AAGM dataset. In first step they have gathered apps and introduced all of gathered apps in real cellphones capture the generated traffic, to identify and labeling to create dataset. In second step feature extraction and selection of suitable features, after that, the data is split into training data and testing data, in ratio of 80 percent and 20 percent respectively.

Last step involves training and testing of data using different machine learning algorithms (Random forest, Random Tree, Decision Tree J48, Regression and K-Nearest Neighbors).

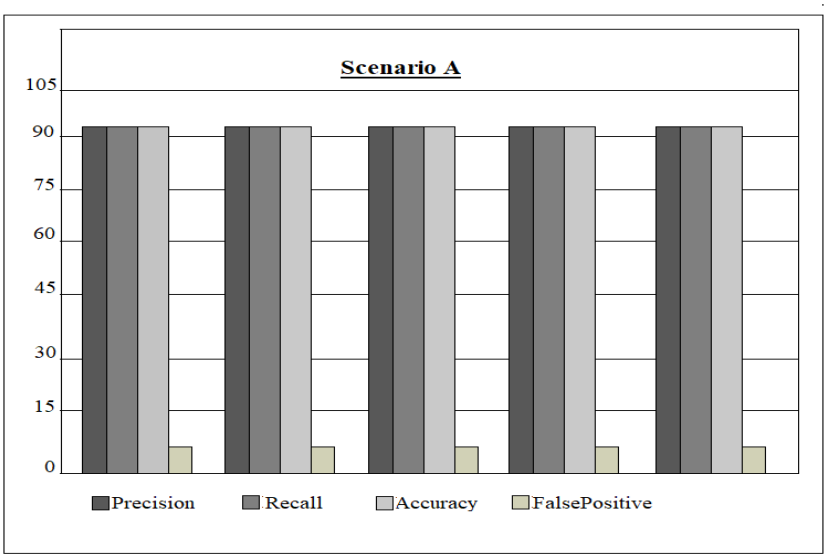

Figure 2. 8 Scenario A for AAGM dataset [37]

Three Scenarios are used in testing and analysis phase:

- Scenario A

For this scenario there are two classes benign and malware (general malware and adware both labeled as malware). Figure 2.8 shows that all the classification algorithms (Random forest, Random Tree, Decision Tree J48, Regression and K-Nearest Neighbors).tree) gave more than 90% precision and accuracy.

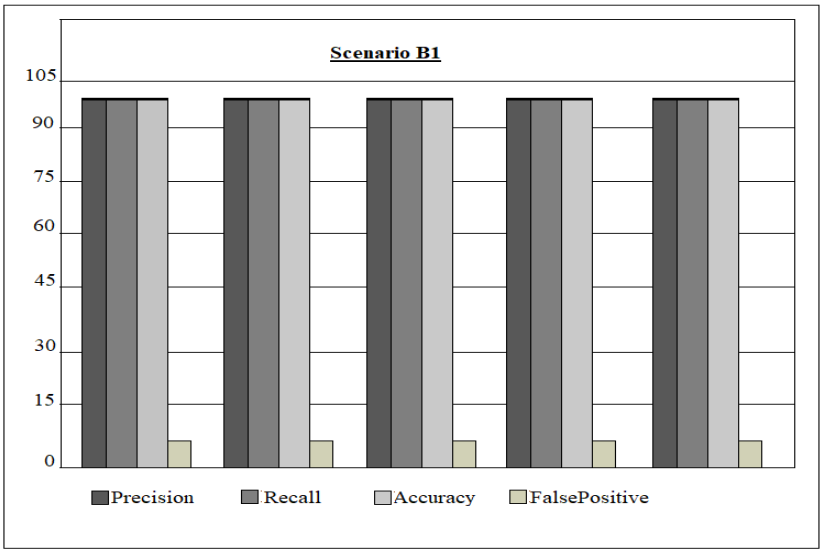

Figure 2. 9 Scenario B1 for AAGM dataset [37]

- Scenario B

Scenario B involves B1 and B2, B1 includes classification between benign and general malware apps using five machine learning algorithms (Random forest, K-Nearest Neighbors, Decision Tree J48, Regression and Random Tree).

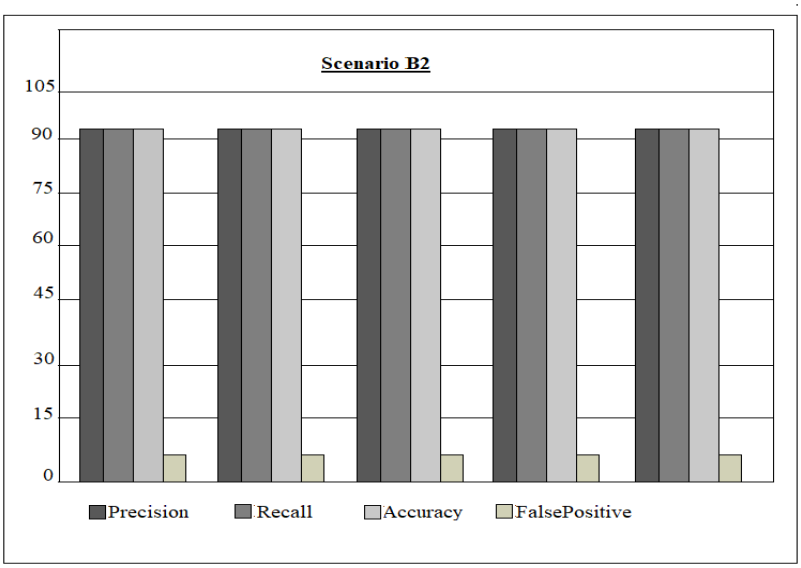

Figure 2. 10 Scenario B2 for AAGM dataset [37]

All the classification algorithms in case of B1 provided around 95 % precision and accuracy. Figure 2.9 shows the scenario B1 for AAGM dataset. Scenario B2 for AAGM dataset is shown

in figure 2.10 which includes the classification of benign and adware apps using above mentioned machine learning algorithms.

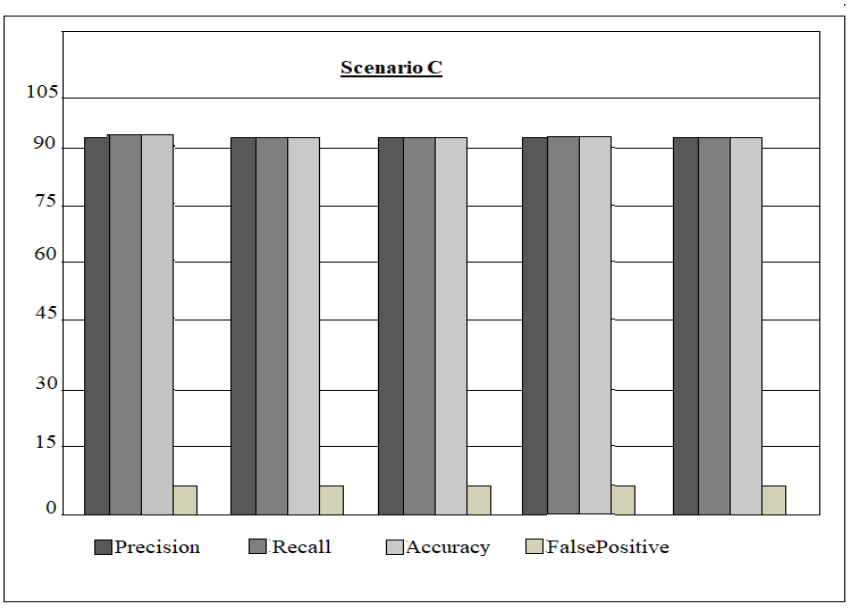

Figure 2. 11 Scenario C for AAGM dataset [37]

The results in case of B2 are nearly matching the results situation B1 and shows the precision of more than 90%.

- Scenario C

Last scenario shows detection and classification results for all categories (general malware, adware and benign apps). For last case in this model we can see that the all the algorithms can classify the malware categories with more than 90% accuracy which is shown in figure 2.11.

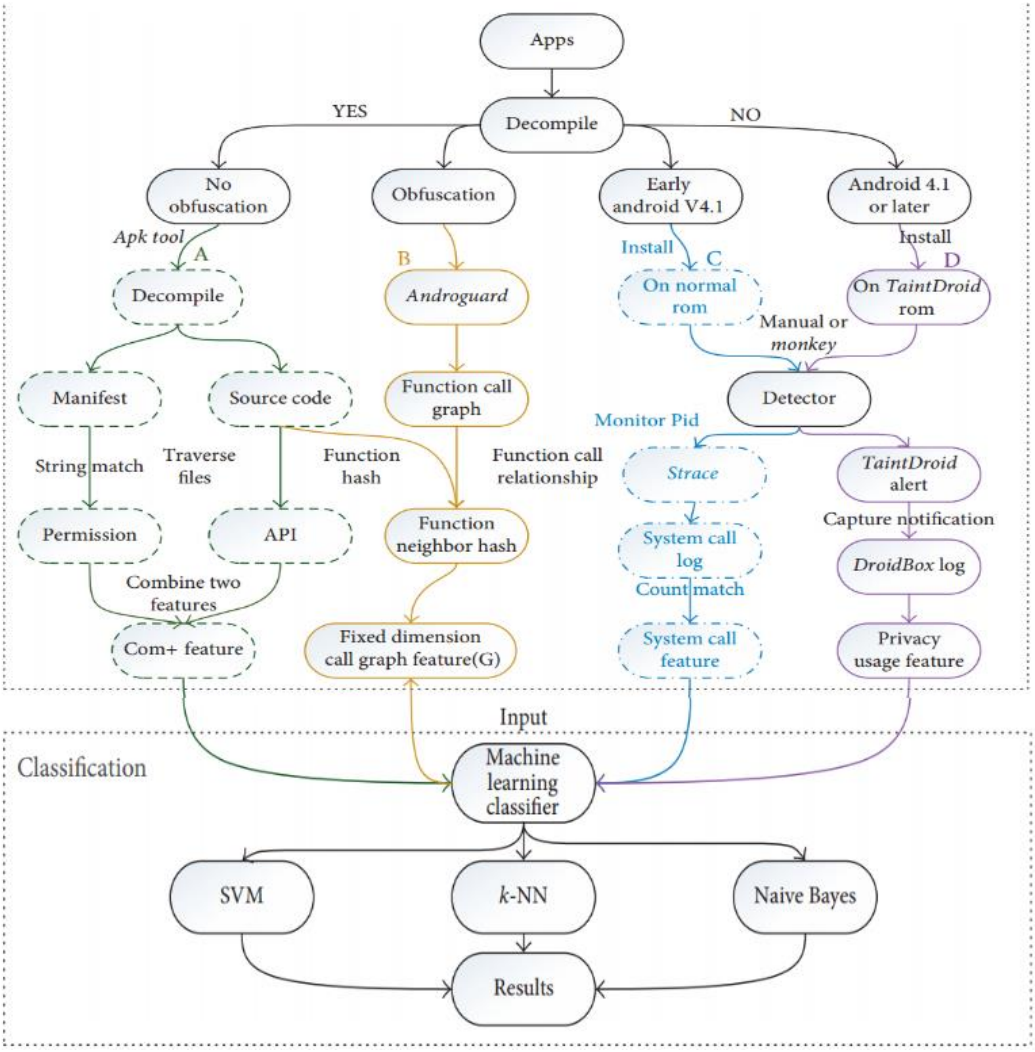

Figure 2. 12 Overview of hybrid detecting scheme [38]

Another paper in 2018 [38] introduced an efficient way to detect android malware by utilizing hybrid technique using several dynamic and static analysis. Hybrid detecting scheme can pick the suitable technique as per the properties of the application with the goal to balance can be accomplished between precision and complexity. Figure 2.12 shows the overview of this scheme. Naïve Bayes, KNN and SVM are used for Classification. Results of older dataset (before 2012) with static analysis using different feature sets for permission features SVM showed the best accuracy with 96.65% followed by KNN and Naïve Bayes with 95.76% and 93.37% respectively. For API features SVM showed highest almost 99% accuracy and KNN and naïve bayes gave 98.42% accuracy and 94.13% accuracy respectively. Combining the

Permission and API features improved the results in for older dataset (before 2012) with KNN, SVM and Naive Bayes providing accuracy of 98.61% 99.21% and 94.41% respectively.

Table 2. 2 Results using strace analysis [38]

Analysis approach	KNN	SVM	Naive Bayes
Results on older dataset (before 2012) using Strace analysis	87.82%	85.41%	90.31%
Results on newer dataset using function call graph analysis	86.56%	86.11%	86.51%
Results of Droid Box analysis on newer dataset	74.23%	66.33%	71.54%

CICAndMal2017 dataset 2017 [39] they have gathered more than 6,000 applications from Google-play in years 2015, 2016, 2017 based on popularity There are 10,854 total samples in this dataset, with 4,354 being malware and 6500 being benign.

Figure 2.13 shows the network architecture for CICAndMal2017 dataset. Authors defined in dynamic analysis, there are 3 phases of data capture that can cause malware behavior during run time.

- Installation: This state occurs after the installation of malware (between one to three minutes)
- Before restart: This state occurs 15 min before restarting phones.
- After restart: This state occurs 15 min after restarting phones.

Dynamic feature execution was achieved by extracting network traffic, permissions, logs, API calls, memory dump and phone statistics from .apk files of applications. On 60% of the training data, the model is trained using 10-fold cross validation, and 40% of the testing data is evaluated using three common classifiers: KNN, Random forest and decision tree. Results for binary classification (between malware and benign) indicate that random forest shows 85.80% precision with 88.30 recall for testing set. KNN shows 85. 40% precision with 88.10 recall for testing set. DT shows 85. 10% precision with 88.0 recall.

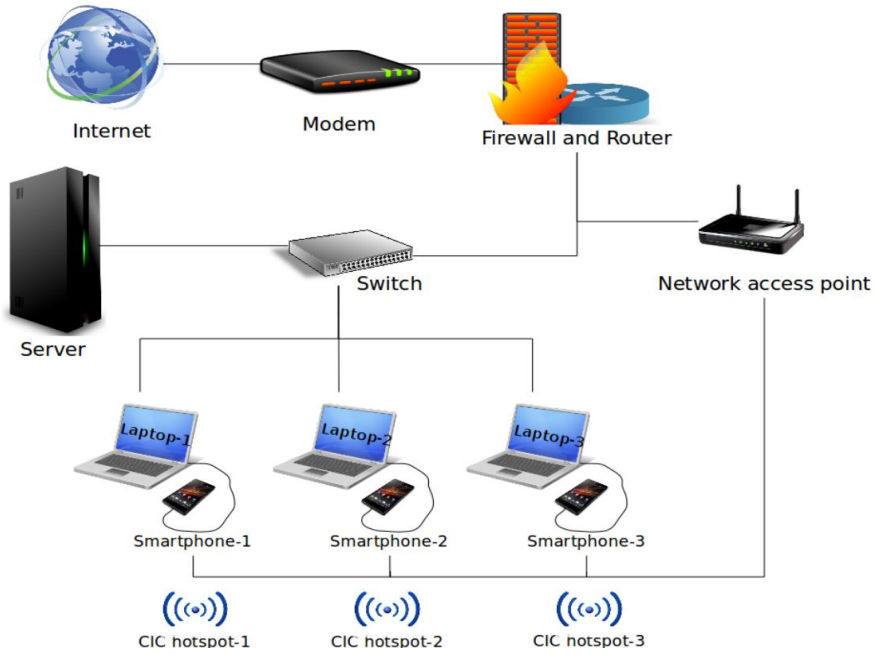

Figure 2. 13 Network architecture for CICAndMal2017 [39]

Malware category classification results show for all three algorithms, precision and recall for testing and training set is between 45 to 50%. In family classification for all three algorithms, precision and recall for testing and training set is between 20% to 27%.

Table 2. 3 Accuracies for CICAndMal2017 dataset 2017 [39]

Dataset	**Training**			**Evaluation (Testing set)**		
Algorithm	**RF**	**KNN**	**DT**	**RF**	**KNN**	**DT**
Binary classification	84	83.6	85.1	85.8	85.4	85.1
Malware Category	46.5	45.7	46.5	49.9	49.5	47.8
Malware Families	22	21.5	21	27.5	27.24	26.6

Researchers structured the accumulated sequence of API method calls of an application gained from DEX assembly and then passed it to a deep learning model for categorization in a study [40] published in 2018. Figure 2.14 shows the approach overview for Maldozer dataset

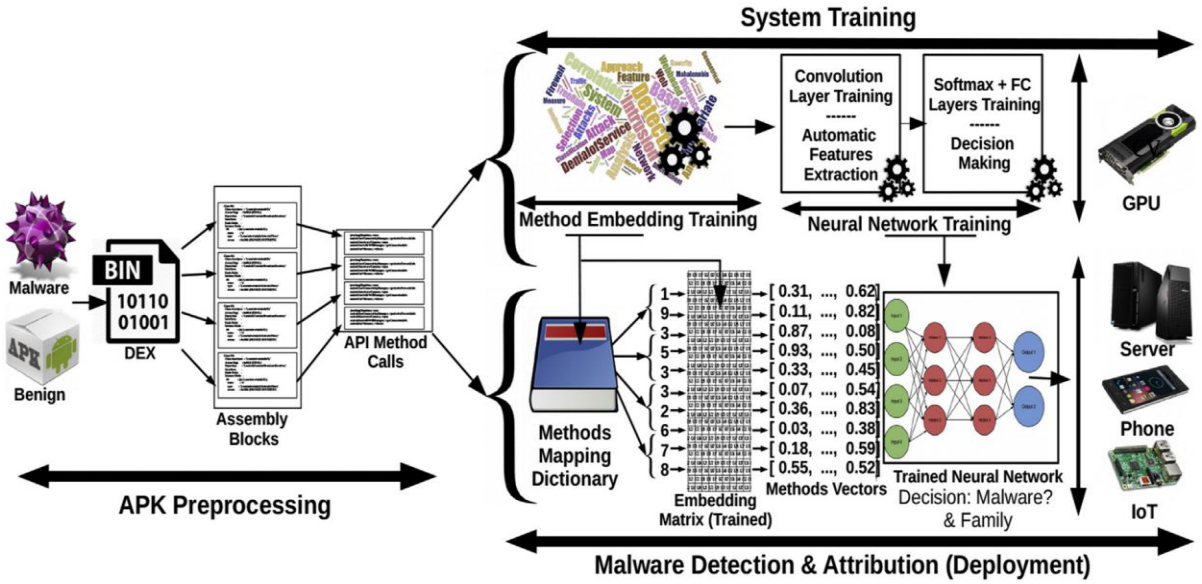

Figure 2. 14 Approach overview for maldozer dataset [40]

They built a dictionary to map every API method call to its unique unified identifier. In pre-processing they created a matrix of unique unified identifier before the training model. The NLP model was used to develop the approach, as many common definitions between NLP model and Android API method calls.

Authors expressed that while in the NLP model we should manage a huge vocabularies and combinations, and we should manage the API method calls sequence and combinations.

They asserted that their model for detection of Android malware is versatile against the over the long haul advancement of API technique calls and their orders. It can be executed on various deployment architectures going from small IoT devices to servers with up to the mark performance.

Table 2. 4 Detection results on datasets [40]

Datasets	F1 (10 folds)	Precision (10 folds)	Recall (10 folds)	FPR (10 folds)
Detection Results on malgenome dataset	99.85%	99.85%	99.85%	0.04%
Detection Results on Drebin dataset	99.22	99.22%	99.22%	0.45%
Detection Results on MalDozer dataset	98.19%	98.19%	98.19	1.15
Detection Results on all dataset	96.3%	96.3%	96.3%	3.19%

In a paper published in 2019 [41] authors used CICAndMal2017 dataset generated by Canadian Institute for Cybersecurity (CIC) [16]. Authors used two layers framework: SBC and DMC

SBC (Static Binary Classification): Static binary classification shown in figure 2.15, is a binary classification model that uses static features to differentiate between benign data and malicious data. The permissions and intent actions are static features retrieved from the App's APK files' ManifestFile.xml and used to build this model. The number of times each extracted permission and intent feature appears in each APK is counted by the model. The Random Forest Learning algorithm is fed these vector data. Vector records are then passed to the Random Forest algorithm.

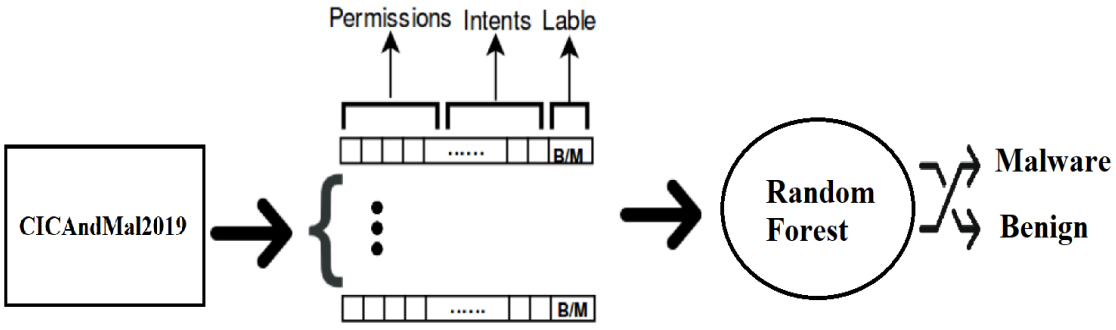

Figure 2. 15 Static binary classification (SBC) [41]

DMC (Dynamic Malware Classification component) : Dynamic Malware Classification layer shown in figure 2.16, authors used network traffic features and API calls to classify the malware into categories (Scareware , Adware, SMSMalware and Ransomware) and then into 39 malware families (AvForAndroid, virusshield , koler, fakeapp, lockerpin, mazarbot, jisut, simplocker, android.spy.277, charger, pletor, , penetho , porndroid, plankton ,mobidash, selfmite, gooligan, svpeng, wannalocker, kemoge, fakejoboffer, dowgin, ewind, avpass, feiwo, AndroidDefender, fakeinst, biige, fakeav, FakeTaoBao, , RansomBO, youmi, shuanetfakemart, , nandrobox, beanbot, jifake, zsone, smssniffer). API calls, according to former researchers, work together to fulfill action tasks on android devices.

In this research, the authors look at the 2-gram sequential linkages in API call collaborations. The diagram depicts an inside look at the DMC component of suggested analytical system. For each instance, the 2-gram sequences of API calls are transformed into a word document. As a consequence, every API call signature is a phrase in a word document made up of multiple phrases separated by a space character. They combine the Before Restart data capturing state and After Restart data capturing state connected to dataset to enhance behavioral data monitoring.

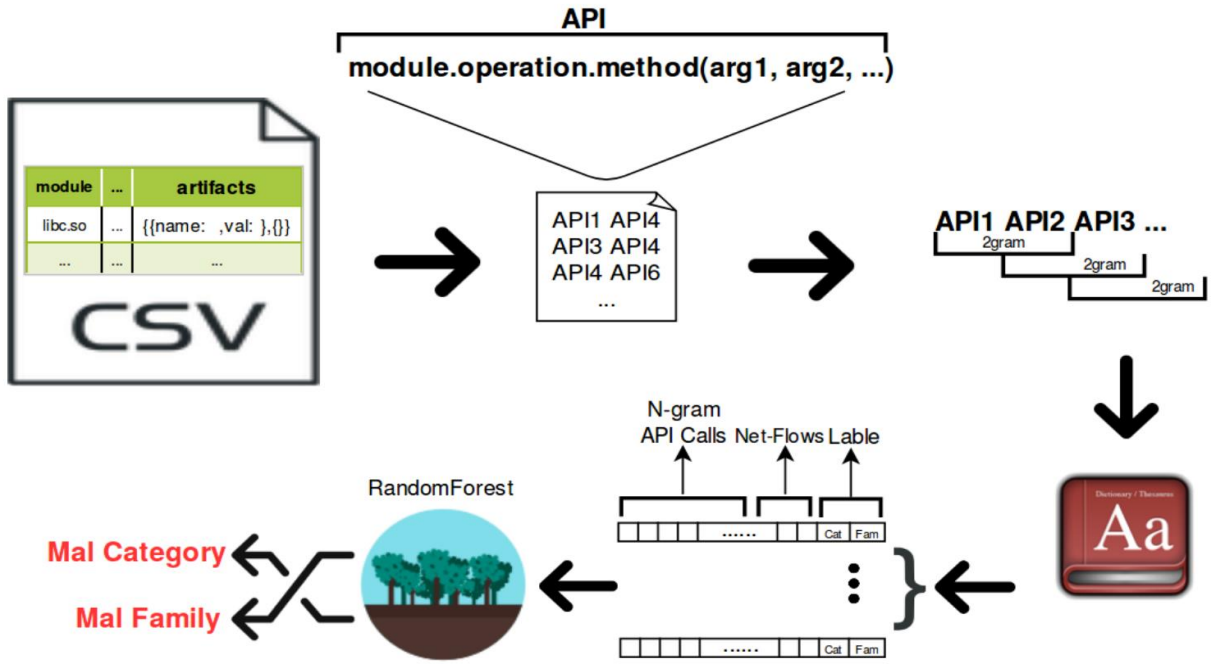

Figure 2. 16 Dynamic malware classification component [41]

The second half of the CICAndMal2017 (CICAndMal2019) includes new feature sets like permissions and intents in the static area and extended API calls in the dynamic section. Authors compared the results for two dataset CICAndMal2017 and CICAndMal2019. Binary classification results between benign applications and malware applications shows almost 10 % increase in accuracy which increased from 85.8 % to 95 %. By merging the dynamic features with 2-gram sequential relations of API requests, increased malware category and family classification performance. In addition, these features were investigated in the two layer malware analysis methodology that was given. Malware category classification between four categories (ransomware, adware, scareware and SMSware) showed the great enhancement in accuracy which jumped to 83.3 % from 49.9%. Malware family classification also showed improvement in accuracy which increased from 27.5 to 59.7%.

Table 2. 5 Results for first and second part of CICAndMal2017 [41]

Scenario:	**Binary**	**Malware Category**	**Malware Family**
Algorithm	Random Forest	Random Forest	Random forest
Precision for CICAndMal2017	85.8%	49.9%	27.5%
Precision for CICAndMal2019	95%	83.3%	59.7%

CHAPTER 3: SURVEY OF MACHINE LEARNING ALGORITHMS

Linear Classifiers, Naive Bayes, K means Clustering, Logistic Regression, Support Vector Machine, , Neural networks, Perceptron, Quadratic Classifiers, Boosting, Random Forest, Bayesian Networks, Decision Tree and so on are among the machine learning algorithms that deal with classification, according to [42].

3.1 Logistic Regression

This algorithm employs a single multinomial logistic regression model using a single estimator and builds on class. In a specific approach, logistic regression generally states where the limit between the classes is, just as how the class probabilities depend on separation from the limit (boundary). When the dataset is greater, this swings nearer to the limits 0 and 1 all the more rapidly. Probabilistic claims like these differentiate logistic regression from other different classifiers. Logistic regression makes more precise, detailed forecasts and can be fitted in a variety of ways; nevertheless, such precise forecasts may be incorrect. There are two approaches to predication ordinary Least Squares regression and logistic regression. Logistic regression prediction yields a binary result [43]. It is one of the most widely utilized technique for applied statistics and discrete data analysis. Linear interpolation is the method used in logistic regression.

3.2 Linear Classifiers

Linear classification methods split input vectors into classes using linear decision boundaries [44]. With linear classifiers, the goal of classification in machine learning is to arrange items with similar feature values into groups. This algorithm accomplishes this purpose, according to [45], by generating a result based on linear combination of features. A linear classifier is commonly used in instances where speed is a priority since it is the quickest classifier. [42]. Linear classifiers also perform well when, if there are enormous dimensions, like in text classification, in which each sample is frequently the number of times a word appears in a document. The degree of convergence of dataset variables, on the other hand, is determined by the margin. The margin, in a nutshell, measures how separable a data is linearly, and thus how simple a classification task is to solve [46].

3.3 Naive Bayesian Networks

Naive Bayesian Networks are made up of directed acyclic graphs with one parent which represents the unobserved node and multiple children which represent observed nodes with a significant assumption of child node independence in the context of the parent node. As a result, this model is dependent on estimating [47]. Even if it was less effective than other more advanced learning algorithms. A large-scale comparison of this classifier with advance and modern algorithms for decision tree induction, instance-based learning, and rule induction on standard datasets found it to be often better to the other learning methods. The attribute independence issue in bayes classifiers was solved using averaged one dependence estimators [48].

3.4 Support Vector Machines

It is similar to multilayer perceptron neural networks. Concept of a margin which separates the classes on either side of a hyperplane is central to SVMs. Increasing the margin and generating the biggest possible distance between the separating hyperplane and the examples on either side of it reduces the predicted generalization error [49].

SVM are based on discovering an optimal hyper plane separating the data of two classes. Figure 3.1 shows the hyperplane separating classes. There are also other methods of selecting the hyper plane which will separate the two classes deprived of any error. In other words, the SVM algorithm produces an output optimum hyperplane that divides or classifies new samples or test data based on the input labelled training data. [50].

The hyperplane is a line in two-dimensional space that divides the plane into two sections. On each side of the hyperplane, we have a separate class. We can say that optimal separating hyper plane is that which is capable of separating the data of two classes and also maximizes the boundary of the hyper plane [69]. The separating hyper plane has the form

$$(W, xi) + b = 0$$

Here b is offset and W is normal to the hyper plane.

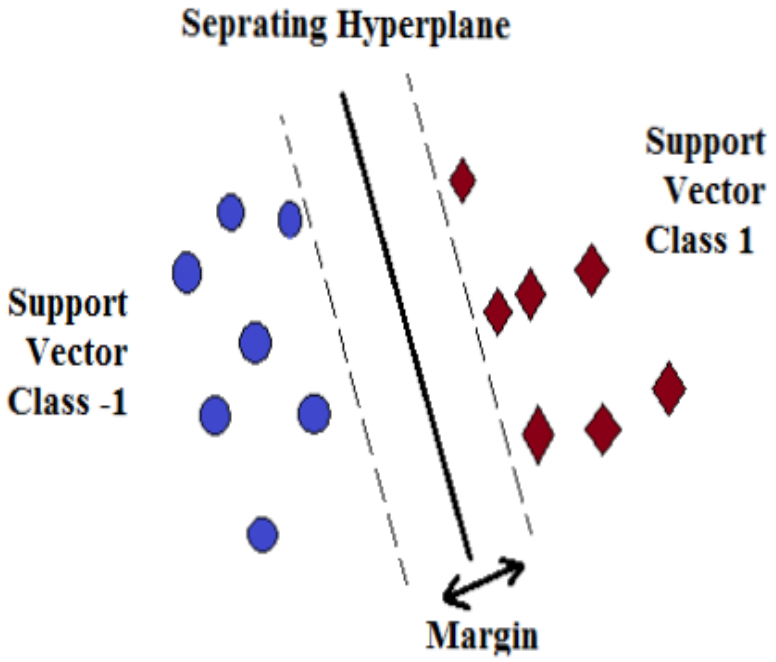

Figure 3. 1 SVM classifier [57]

3.5 Multi-layer Perceptron

Instead of addressing a nonconvex, unconstrained minimization problem as in traditional neural network training [42], the weights of the network are obtained by using linear constraints to solve a quadratic programming problem. The perceptron concept is also used in other well-known algorithms. The perceptron algorithm is utilized to train from a batch of training examples by iterating through the training set till it discovers a correct prediction vector across the board. The test set's labels are then predicted using this prediction rule [49].

3.6 K-Means

K means has been one of the simplest methods for solving the clustering problem, according to [51]. When labelled data is not available, the approach follows a basic and straightforward method for classifying the dataset through predetermined number of clusters fixed a prioriK-Means technique is used. A general strategy for converting sloppy guesses into extremely precise prediction rules. With enough data, a boosting algorithm can provably construct a single classifier with very high accuracy, say, 99 percent, given a weak learning algorithm that can regularly find classifiers at least slightly better than random, say, accuracy Of 55 percent. Neural Networks: According to [51] Neural Networks that can execute many regression or

classification tasks at once, despite the fact that most networks only do one. In the majority of cases, the network would have one output variable, however this may equate to a number of output units in the case of many-state classification issues.

Network architecture, weight and input and activation functions of the unit are all important features of an Artificial Neural Network (ANN). Given that the first two features are constant, the ANN's behavior is determined by the weights' present values

Net training weights are assigned to random values at start, and then samples of the training set are introduced to the net on a regular basis. On input units, the values for an instance's input are set, and the net's output is matched to the instance's desired output. Weights in the net are then modified marginally in the direction of bringing the net's output values up to the desired output values. A number of algorithms can be used to train a network [52].

3.7 Decision Trees

Decision Trees use feature values to rank instances. Every branch refers to the value that node can adopt, and every node indicates a feature in an instance to be categorized. Instances are categorized and arranged based on their feature values, starting at the root node. [49]. A decision tree is a prediction model that translates information about an object decisions about the item's target value in decision tree learning, that which are used in machine learning and in data mining. Classification trees or regression trees are more precise names for such tree models [53]. Post-pruning approaches are commonly used in decision tree classifiers to assess the efficiency of decision trees after they have been reduced by used of validation set. Every node can be removed, and it can be assigned to the highest common class of the sorted training instances. [49].

3.8 Bayesian Network

It depicts the probabilities of a set of variables. Most well example of statistical learning algorithms is Bayesian networks. When compared to neural networks or decision trees, The ability to take into account prior information on a given issue in terms of structural correlations between its features is without a doubt the most intriguing part of Bayesian Network. [9]. However, for datasets with a large number of features, Bayesian Network classifiers are

ineffective. [52]. Prior knowledge on the architecture of a Bayesian network, also known as domain knowledge. It can be expressed in the following ways:

- Declare a node as the root node, implying that it has no children.
- Declare a node as a leaf node, which means it has no children.
- Designating one node as the principal cause or effect of another.
- Declare that a node is not directly connected to any other node.
- Determining the independence of two nodes based on a set of criteria.

3.9 Random Forest

It is intended to form a combination of weak unbiased classifiers which unite their outcomes during the final decision of each item. Classification trees are built with individual classifiers. Collaboration of many supplementing decision trees provides better performance by improving accuracy and gives better generalization [52]. Figure 3.2 shows the working of random forest algorithm.

Distinctive bootstrap sample of training set is used in each tree. Every bootstrap sample is an outcome of drawing with replacing the identical number of samples as in the actual training set. Therefore, approximately 1/3 of items isn't utilized for making a tree and rather is utilized for an out of bag (OOB) estimation of error, and for significance measurement.

An alternate subset of attributes is selected randomly at the each progression of the tree development. While performing a split attribute is used which can made the best distribution of data between the nodes of the tree.

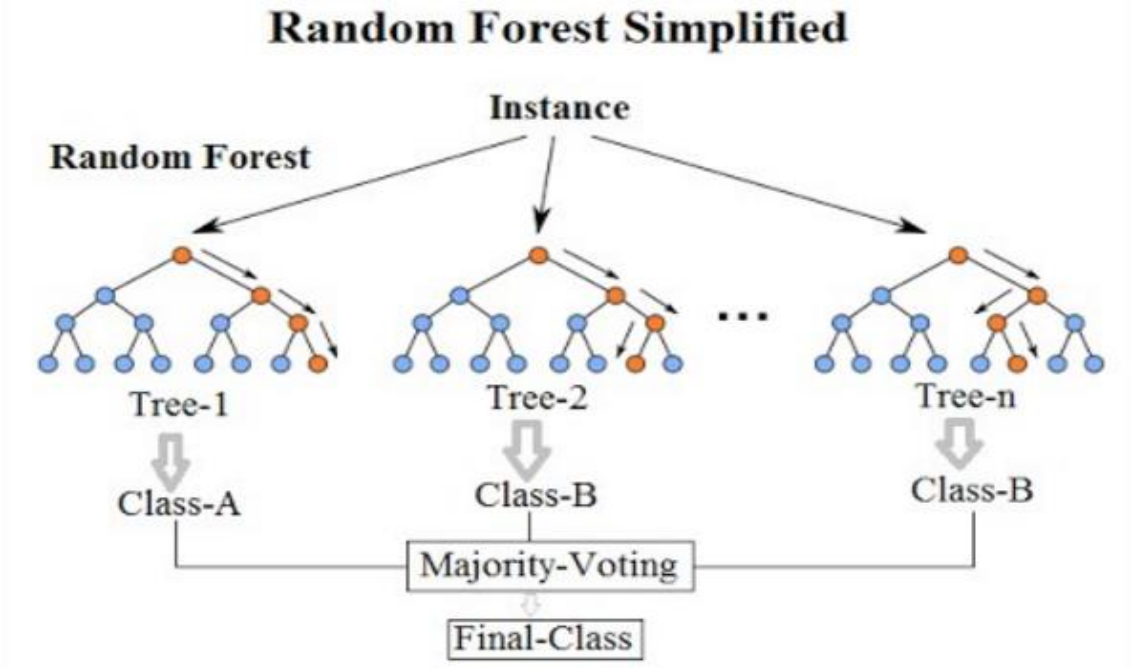

Figure 3. 2 Random forest working [55]

This method is performed until the entire tree is constructed. Developed tree is utilized to classify its OOB objects, and the outcome is utilized for computation of confusion matrices and in acquiring the approximations of the error. New subjects are classified by all the trees in forest, and an ultimate decision is made by vote [56].

The significance of every variable is assessed in the following manner. Initially all subjects are classified. Every tree uses its votes just to the classification of subjects, which were definitely not utilized for its development.

Number of votes in favor of a right class are recorded for each tree. At that point the values of given variable are randomly permuted across subjects, and then the classification process is repeated. Then again for each tree, number of votes are counted for correct class. At that point significance of the variable for the single tree can be characterized as a disparity between the quantity of correct votes cast in favor of original and permuted system divided by number of subjects. Then the significance of the variable is calculated by averaging significance measures

3.10 K-Nearest Neighbors (KNN)

KNN can be used to tackle classification and regression forecasting issues. In industry, however, it is mostly used to tackle classification and prediction problems. KNN uses feature similarity to predict the class of new samples that means new sample will be allocated a category depending on how it resembles the training set [52].

With the help of the steps below, we can grasp how it works.

1) We'll need a dataset to load at the first stage of KNN.
2) Next, we have to choose the k value, i.e. the points closest to it. K can be any integer.
3) Perform the following process for every data point in the test data:

   - Using Euclidean, Manhattan, or Hamming distance methods, compute the distance between every row of training set and test samples. The Euclidean method is the most widely used method for calculating distance.
   - Depending upon the distance, arrange all of them in ascending order.
   - The top K rows of the sorted array will then be chosen.
   - The most common class of these rows will be used to allocate a class to each sample.

Equation below is the formula for calculating Euclidean distance

$$d = \sqrt{(w - w1)^2 + (h - h1)^2}$$

3.11 Features of Machine Learning Algorithms

Machine learning algorithms could be used in a range of fields. When working with multidimensional and continuous features, SVMs and neural networks function substantially better. While working with discrete features, logic based algorithms perform much better. Neural network models and SVMs require a large data set to achieve optimal accuracy, whereas NB may only require a small dataset. KNN is very sensitive to useless features: The way the algorithm operates explains why k-NN is so sensitive to irrelevant features. Furthermore, the existence of non-essential features might make neural network training ineffective, if not impossible. The majority of decision tree algorithms struggle with cases that necessitate diagonal division [52].

The instance space is divided orthogonally to one variable's axis and parallel to the other axes. As a result, after splitting, the resulting regions are all hyperrectangles. When there is multi collinearity and a nonlinear relationship between the input and output features, ANNs and SVMs work well. During the training and classification phases, Naive Bayes uses very little space: The memory required to hold the prior and conditional probabilities is the absolute minimum. The training stage of the basic KNN takes up a lot of space, and the implementation space is as vast as the training space. Contrary to that, because the final classifier is typically a significantly condensed summary of the dataset, execution space for all non-lazy learners is commonly substantially less than training space. Furthermore, whereas rule algorithms cannot be used as incremental learners, Naive Bayes and KNN can. Missing values are usually ignored in determining probabilities, therefore they have no impact on the final choice in Naive Bayes. On the other hand, KNN and neural networks require entire data to function. Decision Trees and Naive Bayes frequently have different operational profiles; while one is extremely accurate, the other is not, and vice versa. The operational profiles of SVM and ANN are comparable. Decision trees and rule classifiers, also have a fairly similar operational profile. Over all datasets, no single learning system can consistently outperform others. Different datasets with varied types of variables and the number of instances influence the sort of method that will function well.

3.12 Feature Reduction

Feature reduction is viable in eliminating insignificant and repetitive information, expanding learning precision, and improving result conceivability. Nonetheless, current rise of dimensionality of information and data poses a challenges feature reduction and selection techniques concerning accuracy and effectiveness [57].

Feature reduction gives a compelling way to resolve this issue by eliminating repetitive and irrelevant information, which can decrease calculation time, improve learning precision, and encourage improved knowledge and understanding for the data or learning model. Figure 3.3 provides a hierarchical structure of approaches used for dimensionality reduction

Dataset with a high dimensions is difficult to classify due to the high computational cost and memory utilization of algorithms. Two approaches are used for reducing dimensionality

One of the most straightforward approaches in feature reduction is through Feature selection; choosing just that information or features that contain the useful data to resolve the specific issue. The benefit of feature selection is that no data regarding the relevance of one feature is destroyed. However, if just a limited number of features is needed and the original set of features is quite diversified, data may be lost since some features need to be excluded during the process of feature subset selection process.

It is a broader method in to map a set of input data onto low-dimensional data that contains a considerable amount of information [58]. The feature space size can be reduced in feature extraction without eliminating a large amount of data from the feature space. The decision between these two methods is influenced by the type of data and application domain.

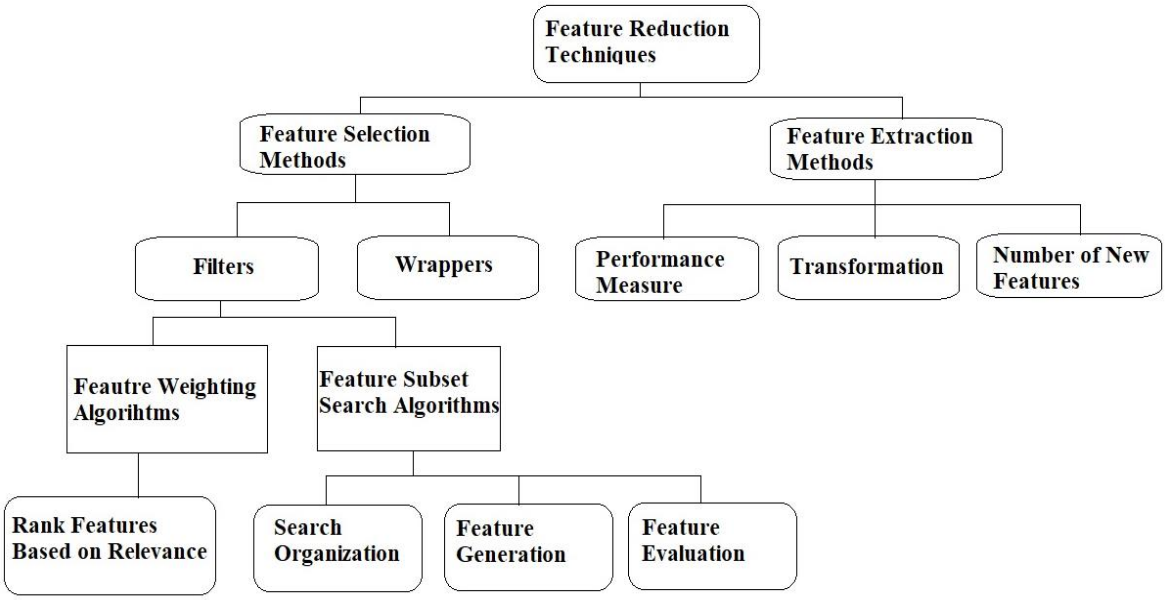

Figure 3. 3 Hierarchical structure of dimensionality reduction approaches [48]

Typically, features are classified as redundant, irrelevant, or useful. During the feature selection process, a subset of data available is picked for the learning algorithm. The best subset is the one with the fewest features that aid precision learning the most [58].

3.12.1 Feature Selection

High dimensional data contains features that may be useless, deceptive, or duplicated, resulting in a larger search space, making it more difficult to interpret data and obstructing the learning process. Choosing the best features which can be used to distinguish classes is known as feature subset selection. It is a computational model that is triggered by a set of relevant criteria. L. Ladha et al [58] conducted an empirical analysis of several feature selection approaches. Wrappers, filters, and hybrid procedures are the three categories of feature selection techniques. Wrappers approaches outperform filter approaches because the feature selection procedure is customized for the classifier to be used. Wrapper approaches, on the other hand, are too expensive to utilize for vast feature spaces due to their high computational costs, and every feature set should be validated using the trained classifier, which slows down the feature selection process. When comparing to wrapper approaches, filter methods have a lower computing cost and are speedier, but they have inferior classification accuracy and suits for high dimensional dataset. Hybrid techniques combines the benefits of filters and wrappers approaches, have recently been developed. It uses a feature subset performance evaluation algorithm as well as an independent test [59]. Feature weighting algorithms and subset search algorithms are the two types of filtering algorithms. It entails some alteration of the original features in order to create more important features. The following is how Brian Ripley [60] described feature extraction: "Feature extraction is generally used to mean the construction of linear combinations of continuous features which have good discriminatory power between classes".

3.12.2 Feature Extraction

It's critical for further data analysis; whether it's data compression, de-noising, visualization, pattern recognition etc., the data must be represented in a way that makes analysis easier. To discover a suitable transformation, several basic approaches have been devised.

3.12.3 Filter Based Feature Selection Methods

These methods normally use the following process. The technique, which is based on a specific searching procedure, begins by searching from a subset that has been provided, which could

be a full, empty, or randomly chosen subset. At that moment, each produced subset is evaluated using a specific metric and compared to the previous best.

This cycle is repeated till the pre-defined criterion is achieved. Subsequently, it yields the most recent best subset. Precisely, Depending on the algorithms used, Evaluation measure and search technique can be different. Few examples of filter based methods are T-testing, regression and principal component analysis (PCA) [61] [62].

3.12.4 Principal Component Analysis

It is an unsupervised approach to minimize the dimensionality of a dataset while maintaining as much data as possible. This is accomplished by identifying a new set of features known as components, which are uncorrelated composites of the original features [63]. It is a frequently used feature extraction technique in pattern recognition and computer vision applications.

PCA isn't ideal in a lot of situations: For example, if all of the PCA components have a high variance, there really is no universal stopping criteria that allows you to remove a specific 'k' Principal Components while ensuring that no significant data loss occurs during compression. Second, it isn't appropriate for some classification issues. If the data is divided into two classes, but the variance within-class is much higher than the between-class variance, PCA may results in deleting the data that distinguishes your two classes. In other terms, if the data is noisy and the variance between the means of the two groups is greater than the variance between the noise components, PCA will retain the noise components while discarding the distinguishing component (As PCA is unsupervised it is expected). Data in the bottom space becomes indistinguishable as a result of this. Some of the cases in which PCA does not work properly are

- When there is an imbalance in the data.
- When there is a lot of noise in the data it does not gives good results.
- When the significance of certain of the independent variables is high
- When the variance inside a class is greater than the variance across classes.
- When there are numerous independent variables with a given number of classes, PCA ignores the classes and instead looks for an axis with a high variance, resulting in data loss.

3.12.5 T-Test

The t-test technique is used to check if there is a critical variation between two class's or group's means. It assists in determining if the two groups or classes are from same pool and only different due to some error or the groups really have some significant difference [64].

Three fundamental factors help in determining if the groups really have true difference

1. There is a very less chance that difference is because of sampling error or by chance if the sample group is large,
2. There is a very less chance that difference is because of sampling error if there is a big difference between the means of two groups.
3. There is a very less chance that difference is because of sampling error if the variance among participants is very small.

$$t = (x - \mu)/(s/\sqrt{n})$$

Here S is standard deviation, x is sample mean, μ is hypothesized mean and deviation n is sample size.

P value tells how likely we might have gotten these outcomes by some chance, if indeed the null hypothesis were valid. Usually, if there is under 5% ($P < 0.05$) possibility of getting the observed differences by some chance, we reject the null hypothesis and state we found a significant variation between two sample groups.

CHAPTER 4: METHODOLOGY

4.1 Data Set Description

CICInvesAndMal2019 [14] dataset is generated by Canadian Institute for Cybersecurity (CIC) they have gathered 10,854 samples out of which 4,354 were malware and 6,500 were genuine benign apps from a variety of places, including the previous researchers [65], [37], [66], VirusTotal service [67], the Contagio security blog [68]. But due to the following two factors, only been able to install 5,000 of them (malware 429 and benign 5,065):

1) Sample Error: Most of the samples were of poor quality, with Dex issues, unsigned, and corrupt apps among them. All apps must be signed in order to be installed on an emulator or a genuine smartphone.
2) Inconsistent malware labelling: Inconsistent labelling among anti-virus companies and researchers causes uncertainty and makes re-organization time-consuming. To solve this issue they compared malware labelling from many anti-virus companies and then adopt the majority numbers among the most common label of the malware family in question.

4.1.1 Configuration Architecture

Figure 4.1 shows three laptops are connected to three smartphones in the following architecture

1) Laptop 1 with android phone-1: a NEXUS 5 mobile device connected to a 64-bit Ubuntu 16.04 laptop with an Intel CPU and 8GB RAM.
2) Laptop 2 with android phone-2: a NEXUS 4 mobile device connected to a 64-bit Ubuntu 16.04 laptop with an Intel CPU and 8GB RAM.
3) Laptop 3 with android phone-3: a NEXUS 5 mobile device connected to a 64-bit Ubuntu 16.04 laptop with an Intel CPU and 8GB RAM.

In dynamic analysis, three phases of data gathering was developed that potentially activate malware activity at run time during execution [39].

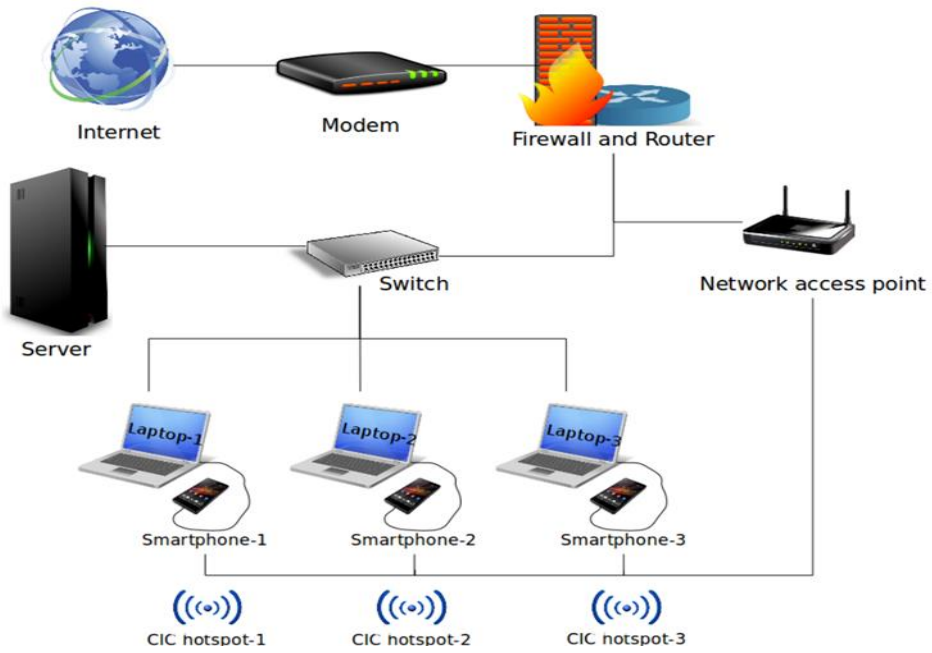

Figure 4. 1 Configuration architecture [39]

- Installation: Occurs after the installation of malware (between one to three minutes)
- Before restart: This situation lasts for 15 minutes before the phones are restarted..
- After restart: Occurs 15 min after restarting phones.

4.1.2 Malware Activation Scenario

To better identify malware activity and features, Taxonomy was designed to establish a complete data capture scenario. There are 20 types of attacks (A1-A20) and four forms of command and control communications (C&C) in the taxonomy (C1-C4) which is shown in figure 4.2.

- User Profile: For each smart phone, three user profiles were used. Each user has a valid email address and phone number for Gmail, Facebook, Skype, and Whatsapp.
- User interaction scenarios: Specific scenarios were generated for each malware category in order to provide a consistent data output.

1) Benign: “Browse Internet, Send Message, Enable GPS, Make Call”.

2) Adware: “Browse Internet, Send Message, Enable GPS, Make Call”.

3) Scareware: “Browse Internet, Send Message, Enable GPS, Make Call, Click/follow popup”.

4) Ransomware: “Browse the Web, Send text Messages, Enable GPS (Global positioning system), Make Calls, Click/Follow Popups, Established the 4 digit PIN and lock the mobile, select and Interact with any pop - up Text, Save contacts (more than 10) in the contact List, Save the following documents for both internal and external storage: doc, txt, jpeg, docx, jpg, mkv, gif, png, bmp, gif, pdf, avi”. [39]

A1	Steal information (contacts & SMS messages, credit card credentials)	A13	Locking the device
A2	Harvest information (text messages, mobile accounts and bookmark history)	A14	Encrypt user files Alp: Modify
A3	Send SPAM SMS or Phishing to the contact list	A15	Modify the contents,(SD card)
A4	Make calls in background	A16	Change PIN number
A5	Exploit root	A17	Use of Video Camera
A6	Inject malicious code	A18	Voice message to unlock screen
A7	Download malicious software	A19	Fake login screen
A8	Premium subscription	A20	Change the wallpaper
A9	Direct user to malicious web site	C1	C&C (command and control communication)servers
A10	Prevalence of malware (uninstall AVs, encrypt source, load dynamic code)	C2	Start after time delay
A11	Geographic location attack	C3	Update with new features
A12	Send shortened URLs	C4	Using TOR

Figure 4. 2 Categories of attacks (A1-A20) and four types of command and control communications (C&C) (C1-C4) [39]

5) SMSware: Browse Internet, Send Message, Enable GPS, Make Call, Install AV(AVG, Avast, BitDefender), Save more than 10 contacts in the contact list.

4.2 Dataset

Dataset contains two layers

First layer: For classification between malware and benign ware apps.

Second layer: For classification of malware into categories and families.

4.2.1 First Layer

First Layer dataset has samples of benign ware and malware apps each sample have 8114 features. First layer contains intent actions and requested permissions as static features which are obtained from ManifestFile.xml of Application's APK files.

4.2.2 Second Layer

Second layer dataset has 919 samples of 4 malware categories and 39 families including:

1. Ransomware
2. Adware
3. Scareware
4. Smsware.

Table 4. 1 Description of Dataset [39]

Category	Family	Year	Total Collected	Total Captured
Adware	Dosvgin	2016	50	10
	Mond	2017	50	10
	Fens	2016	100	15
	Goohgan	2016	43	14
	Kemoge	2015	35	11
	Koodom	2017	50	10
	Mobidash	2015	32	10
	SeItmrte	2014	6	4
	Shuanet	2015	24	10
	Younn	2015	50	10
Ransomeware	Charger	2017	150	10
	Bsut	2017	150	10
	Koler	2015	135	10
	LockerPm	2015	150	10

	Smplooker	2015	146	10
	Mawr	2014	112	10
	Ponillrold	2016	113	10
	RansomB0	2017	100	10
	Svpeng	2014	120	11
	Wannalocker	2017	18	10
Scareware	AndroidDefender	2013	150	17
	AndroldSpy.277	2016	150	6
	AV for Android	2015	83	10
	AVpass	2013	150	10
	FakeApp	2015	150	10
	FakeApp.AL	2015	150	11
	FakeAV	2013	150	10
	FakebabOffer	2013	9	9
	FakeTaoBao	2013	150	9
	Penetho	2012	150	10
	WrusSbield	2014	150	10
SMSware	BemBot	2011	85	10
	Biige	2012	147	11
	FakeInst	2013	150	11
	FakeMart	2012	63	10
	FakeNotify	2012	150	10
	Make	2012	149	10
	Mmarbot	2016	22	10
	Nandrobox	2012	150	11
	Plankton	2011	150	10
	SMSsniffer	2011	53	9
	Zsone	2011	150	10
Total			4354	429

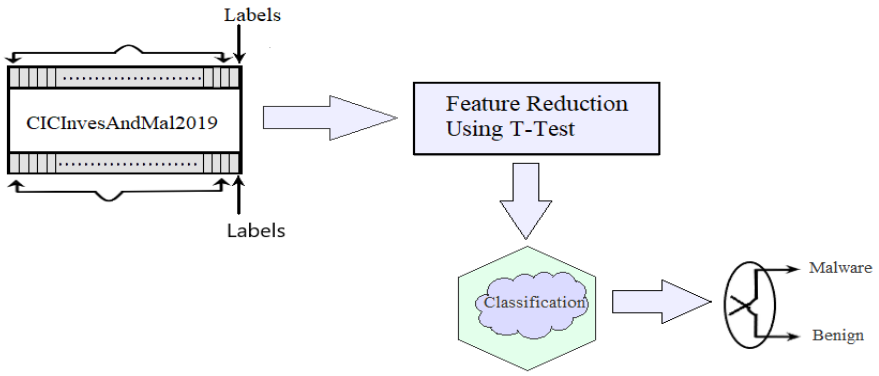

Figure 4. 3 Inside view of first layer methodology

4.3 First Layer Methodology (Malware and Benign)

Originally there were 8114 features in first layer which were to classify between malware and benignware apps. We have reduced the features in first layer and used T-testing for feature selection. Figure 4.3 provides the inside view of first layer methodology

For better look in to features we have took variances for all features. Figure 4.4 shows the Variance plot. Feature reduction in first layer really helped in reduced computation time and improved the results. To get the best features after applying t-testing we have only picked the features which have P values less than 0.05. In this case feature were reduced to 398 (with p values less for classification we used we used three machine learning algorithms

- KNN
- SVM
- Random Forest

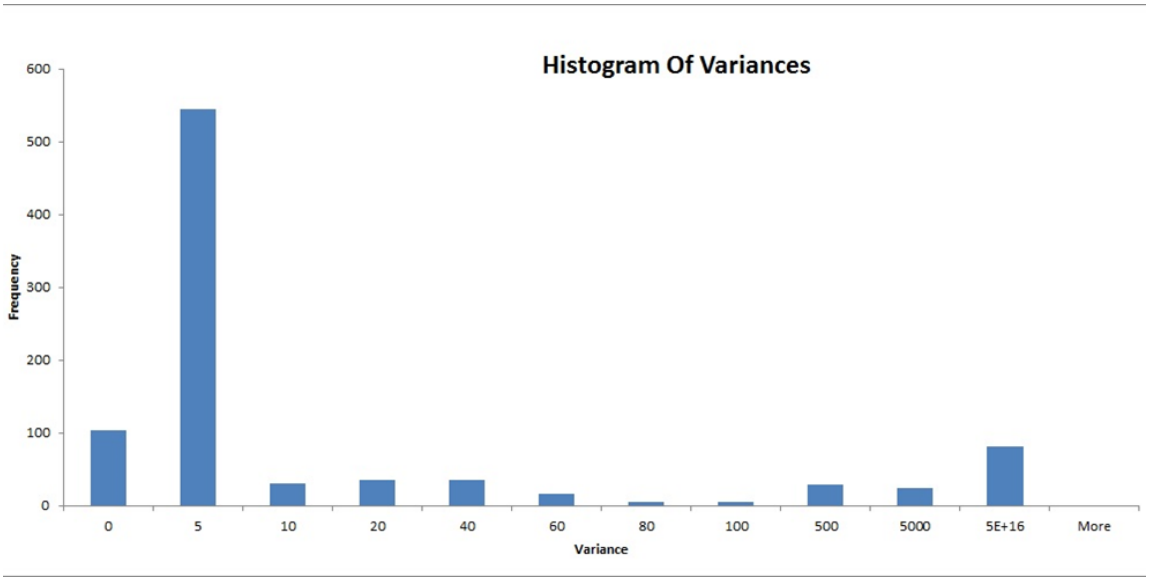

Figure 4. 4 Histogram of variance for first layer

4.4 Second Layer Methodology

4.4.1 Category Classification

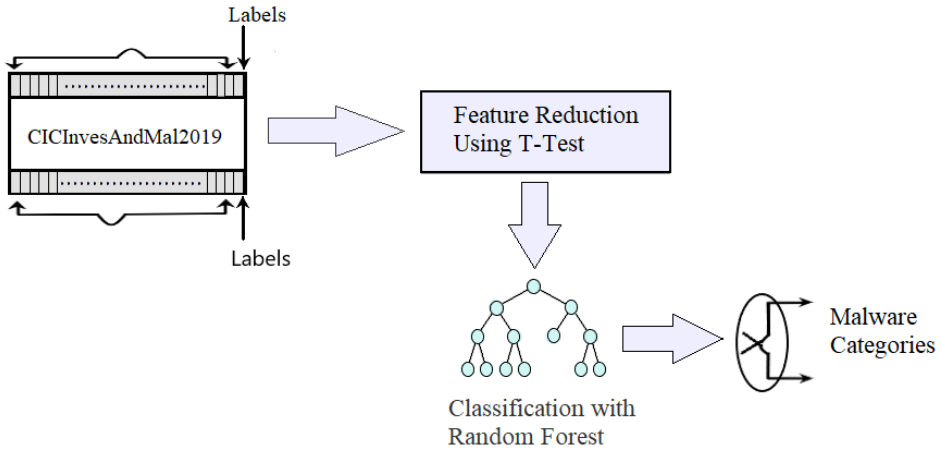

Figure 4. 5 Inside view of malware category classification methodology

There were 919 features for category classification which we had to classify between 4 malware categories (Ransomware, Adware scareware and Smsware). We have reduced the features in second layer and used T-testing for feature selection and then applied Random

Forest algorithm for classification. Figure 4.5 provides the inside view of category classification methodology

For better look in to features we have took variances for all features. Figure 4.6 shows the Variance plot.

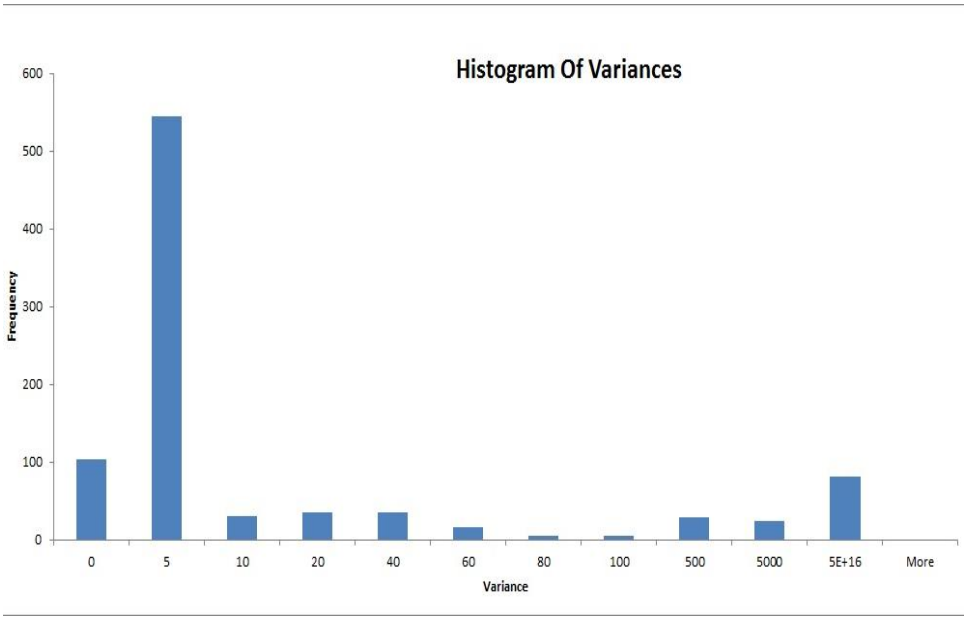

Figure 4. 6 Histogram of variance for second layer

As it was not a binary class problem so we applied the T-testing stepwise. Figure 4.7 shows the feature reduction scheme. First we took P values as follows

- T-test was applied on Ransomware vs all categories (Adware, scareware and smsware) and features with p value less than 0.05 are saved Which are 45 features
- At second stage after removing all samples of Ransomware (already used in stage one) again t-test was applied on Adware vs Rest categories (scareware and smsware).
- P values for Adware vs rest (scareware and smsware) are obtained and features with p value less than 0.05 are saved, which are 69 features
- In third stage after removing samples of Ransomware and Adware again T test was applied on scareware vs smsware.
- Features with p value less than 0.05 were saved, which were 158 features
- In next stage we took union of all those features (45, 69 and 158 from above) which gave us the total of 205 features.

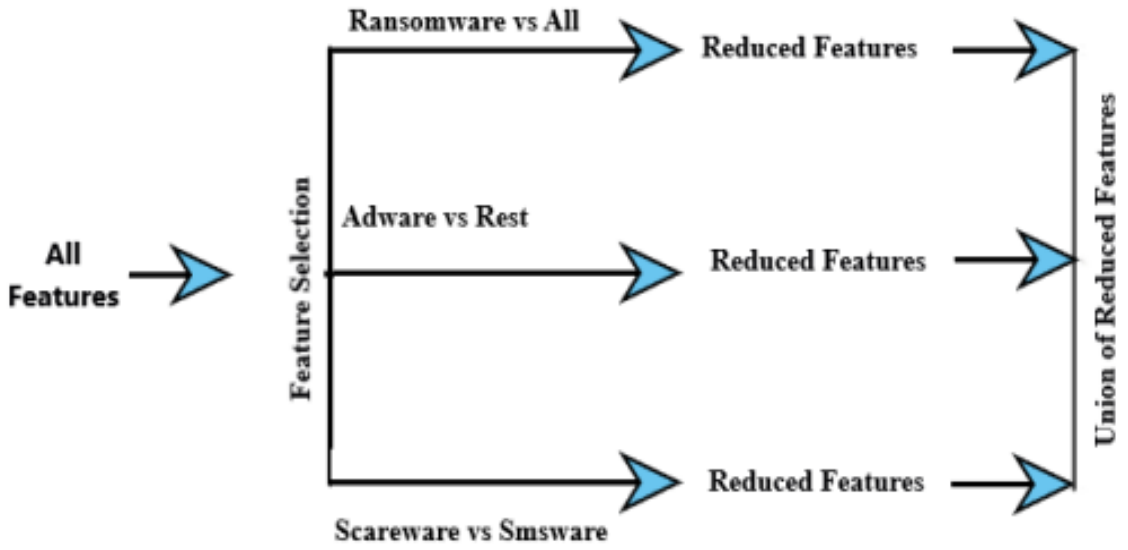

Figure 4. 7 Feature reduction scheme

4.4.2 Family Classification

There were 919 features for family classification, we had to classify 39 malware families. We have reduced the features and used T-testing for feature selection and then applied Random Forest algorithm for classification. Figure 4.5 provides the inside view of second layer methodology.

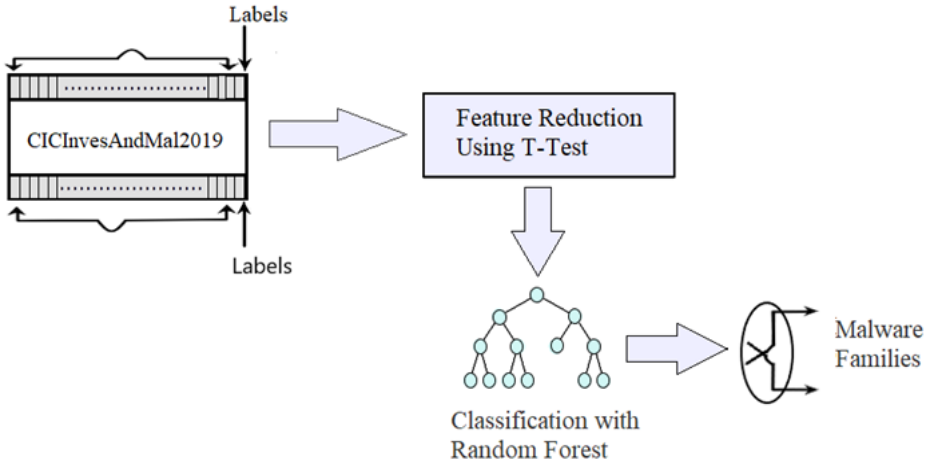

Figure 4. 8 Inside view of Malware Family Classification

4.5 Category Classification with Nested Feature Selection within Random Forest Algorithm

In this algorithm we have updated the conventional random forest by incorporating feature selection into the classification algorithm. At each step it will first use the t-testing for feature selection and then will classify the subjects with Random Forest algorithm. Figure 4.8 shows the structure of algorithm for category classification with nested feature Selection within Random Forest.

To classify the malware into categories first we have applied feature selection, only selected the features with p values less than threshold and saved those features for training and testing files.

4.5.1 Feature Selection

- At first stage p values for Ransomware vs all categories are obtained and best features with p value < 0.4 are saved for training and testing files.
- At second stage after removing all samples of Ransomware (already used in stage one) p values for Adware vs rest (scareware and smsware) are obtained and features with p value < 0.4 are saved for training and testing files.
- In third stage after removing samples of Ransomware and Adware again applied feature selection by obtaining p values between scareware vs smsware. Features with p value < 0.4 are saved for training and testing files.

So, after three phases of feature selection, random forest uses three sets of features in tree-based random forest classification.

4.5.2 Training

At training stage the data was trained separately for three sets of features

- First for ransomware vs all (first stage features)
- Adware vs rest (2^{nd} stage features)
- Sacreware vs smsware (3^{rd} stage features)

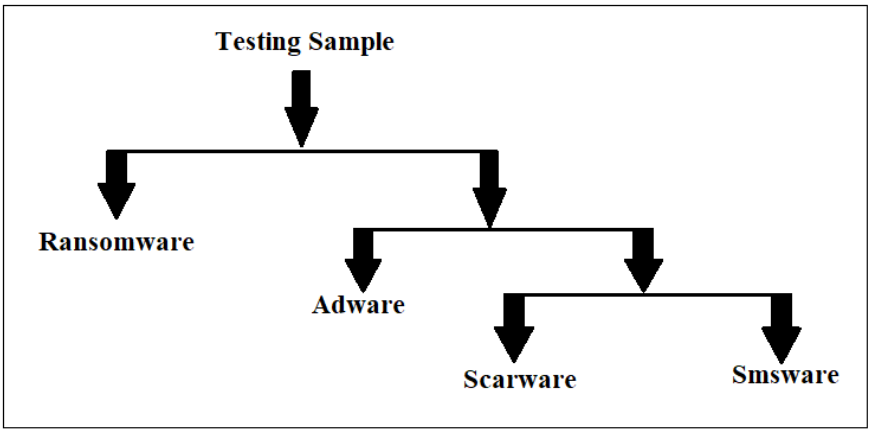

Figure 4. 9 Structure of algorithm for category classification with nested feature selection within random forest

4.5.3 Testing

At testing stage a tree based random forest algorithm is applied on testing data. There were 116 samples of testing data and at each time the algorithm takes only one sample from the testing data. Each sample passes through tree based algorithm in following way:

Whenever testing sample comes the algorithm passes it to stage one of algorithm which classifies the sample between Ransomware Vs All categories (Adware, Scareware vs smsware) if the algorithm predicts the sample as Ransomware it saves the prediction in the prediction array and bypasses the other two stages and next sample comes. If the algorithm predicts that sample is not Ransomware it does not save the prediction and passes that sample to stage two of algorithm which classifies the sample between Adware vs rest (Scareware vs Smsware)

If the algorithm predicts at stage two that sample is adware it saves the prediction in the prediction array and bypasses the 3rd stage.

If in second stage algorithm predicts that sample is not adware it does not save the prediction and passes the sample to last stage which classifies the sample between Scareware vs Smsware and saves the prediction in the prediction array.

So 116 samples of testing data passes through three stages of tree based random forest algorithm one by one and accuracy is calculated with comparing the prediction array with testing labels in the end.

CHAPTER 5: EXPERIMENTATION AND RESULTS

5.1 First Layer (Malware and Benign)

We got good results with Random Forest, KNN and SVM algorithms in binary classification (First layer). Random forest showed the best accuracy with 95.1% for binary classification. Table below shows the Results for different algorithms we used.

Table 5. 1 Models and accuracy results for first layer

Malware binary classification with feature selection			
Method	**Accuracy**	**Precision**	**Recall**
SVM	92.1%	88.7%	78%
KNN	93.5%	-	-
Random forest	93.8%	92.3	82.5%
Random forest Built in	95.3%	96.4	83.75
Neural Network	93.1%	-	-

5.2 Second layer (Malware Category)

We got the best accuracy of 84.48 for Malware category classification using Random forest algorithm table below shows the classification results.

Table 5. 2 Models and accuracy results for category classification

Malware category classification			
Method	**Accuracy**	**Recall/Sensitivity**	**Precision**
Random Forest built in	83.62%	83.62%	83..62%
Random Forest	84.5%	84.5%	84.5%
Neural Network	54.3%	-	-

5.3 Category Classification with Nested Feature Selection within Random Forest Algorithm

For this algorithm of category classification with nested feature selection within random forest algorithm we got fairly good results for category classification with accuracy of 82 %.

Table 5. 3 Malware category classification with feature selection within Random Forest Algorithm

Malware category classification with feature selection		
Accuracy	**Recall/Sensitivity**	**Precision**
81.9 %	81.9%	81.9%

5.4 Family Classification

There are 39 families of malware and we have 189 samples of training and 116 samples of testing data in this part. So very few samples are there of each family for training and testing. There were 919 features, we have used random forest classifier to classify with full 919 features. We Also used those 205 reduced features which we reduced during category classification, as for 39 families it was difficult to take p values for each.

Table 5. 4 Family classification with feature selection nested tree

Family classification with feature selection Nested tree		
Method	**Accuracy**	**Error rate**
Random Forest built	66.38%	0.336
Random Forest built	62.07%	0.379
Neural Network	34.5%	0.65

5.5 Impact of Feature Reduction and T-Testing

CICInvesAndMal2019 dataset has 8114 features for malware detection more number of features means more complexity and more calculations for the classification algorithm. As we look closely into the dataset there is a lot of redundant and ineffective information which can add complexity and can effect results as well. Many of the features does not have values and have very little variance. T-test assumes that:

1. Data is quantitative.
2. The differences are unrelated to one another.
3. They're evenly distributed.
4. Each group being compared has a same level of variance

T-test is used when the sample parameters (standard deviation and mean) are unknown. CICInvesAndMal2019 dataset is quantitative with features independent of each other so to remove the redundant and ineffective features (with low variance) we used T-testing to reduce the features. We have successfully reduced the features and after reduction the classification models gave fairly good results with less computations and in less time.

Table 5. 5 Impact of Feature Reduction

Type	Before Feature Reduction	After Feature Reduction
Binary Classification	8114	398
Category Classification	919	192
Family Classification	919	192

PCA isn't ideal in this situation like if all of the PCA components have a high variance, there really is no universal stopping criteria that allows you to remove a specific 'k' Principal Components while ensuring that no significant data loss occurs during compression. If the data

is divided into two classes, but the variance within-class is much higher than the between-class variance, PCA may results in deleting the data that distinguishes your two classes. In other terms, if the data is noisy and the variance between the means of the two groups is greater than the variance between the noise components, PCA will retain the noise components while discarding the distinguishing component (as PCA is unsupervised it is expected).

5.6 Results Using Random Forest Model

In this scenario, the random forest classification model worked best for us. A single pass on the dataset, like a decision tree, is unlikely to yield the best forecast. If we ran various decision tree algorithms using different subsets of training data and then compared the results to discover which was most frequently given decision in each case.

We can do many runs on the dataset with different subsets incorporating all of the training dataset using the decision tree method in this manner. Finally, we trained a number of decision trees, each of which returned a prediction for each observation in the dataset. The forecasts provided by each model may not be accurate on their own, but when combined, they will be closer to the mark on average.

CHAPTER 6: CONCLUSION & FUTURE WORK

6.1 Conclusion

In the age of smartphones and latest technology it is very important to work on security of common users. There are many datasets available publicly. On latest dataset of CICInvesAndMal2019 generated by Canadian Institute for Cybersecurity (CIC) results we got here are encouraging.

Sometimes more features or curse of dimensionality create problems in getting good results, reducing features using t-testing has worked in this case which solves the problem of computation as well. In addition, the random forest has shown fairly good results with modified algorithm.

PCA isn't ideal in a lot of situations. When the significance of certain of the independent variables is high or when the variance inside a class is greater than the variance across classes, it does not gives best results. When there are numerous independent variables with a given number of classes, PCA ignores the classes and instead looks for an axis with a high variance, resulting in data loss.

In binary classification the use of t-testing with random forest classifier provides promising results as the algorithm could classify the benign and malware apps with good accuracy. There were total of 8114 features in the dataset which we have reduced with the help of t-testing and after feature reduction the features are reduced to 398 features. We have also used the K nearest neighbor and SVM for classification for binary classification.

Our model can classify the malware categories after feature reduction with good precision and recall value. In Category Classification with nested feature Selection within Random Forest Algorithm, we have modified the conventional random forest algorithm by using feature selection inside the classification algorithm. At each step it uses the t-testing for feature selection and then will classify the subjects with Random Forest algorithm. This model shows the precision of 82 %.

In family classification our model shows improved results after feature reduction but due to less samples for training and testing we could not achieve the best results as there are 39 families and for each family there are very few samples to train and test.

Overall the model provides the considerable results for binary, category and family classification which can be improved further using deep neural network

6.2 Contribution

- Reduction of features from 8114 to 398 features in binary classification with improved results.
- Reduction of features for category classification from 919 to 192 features with improved results.
- Modified the conventional random forest algorithm which used nested feature selection within random forest Algorithm.
- Featured reduction in family classification with improvement of accuracy

6.3 Future Work

We have used random forest which gives the encouraging results in all classification problems (binary classification, category classification and family classification). There are few samples for family classification for each family which effects the results with increase in the number of samples in family classification the results can improve considerably.

We can work with nested feature reduction in combination with deep neural network so that the features which are important for each category or family can be used and the redundancy or useless features can be removed in process of classification.

References

[1] Smartphone users 2026 | Statista. (2021). Retrieved 18 August 2021, from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[2] Chatterjee, S., Paul, K., Roy, R., & Nath, A. (2016). A Comprehensive Study on Security issues in Android Mobile Phone—Scope and Challenges. *International Journal of Innovative Research in Advanced Engineering*, *3*(3), 62-72.

[3] Infographic: The Smartphone Platform War Is Over. (2021). Retrieved 18 August 2021, from https://www.statista.com/chart/4112/ smart phone-platform-market-share/

[4] Market Share Alert: Preliminary, Mobile Phones, Worldwide, 1Q17. (2017). Retrieved 18 August 2021, from https://www.gartner.com/en/documents/3664335/market-share-alert-preliminary-mobile-phones-worldwide-1

[5] Mobile Operating System Market Share Worldwide | StatCounter Global Stats. (2021). Retrieved 18 August 2021, from https://gs.statcounter.com/os-market-share/mobile/worldwide

[6] Kelly, G. (2021). Report: 97% Of Mobile Malware Is On Android. This Is The Easy Way You Stay Safe. Retrieved 18 August 2021, from https://www.forbes.com/sites/gordonkelly/2014/03/24/report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-stay-safe/

[7] Number of malware attacks per year 2020 | Statista. (2021). Retrieved 18 August 2021, from https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/

[8] 2021 Cyber Security Statistics Trends & Data. (2021). Retrieved 18 August 2021, from https://purplesec.us/resources/cyber-security-statistics/

[9] 2021 Cyber Security Statistics Trends & Data. (2021). Retrieved 18 August 2021, from https://purplesec.us/resources/cyber-security-statistics/

[10] Number of ransomware attacks per year 2020 | Statista. (2021). Retrieved 18 August 2021, from https://www.statista.com/statistics/494947/ransomware-attacks-per-year-worldwide/

[11] Countries with highest ransomware infection rates 2018 | Statista. (2021). Retrieved 18 August 2021, from https://www.statista.com/statistics/226492/leading-ransomware-infected-countries/

[12] Mobile malware evolution 2020. (2021). Retrieved 18 August 2021, from https://securelist.com/mobile-malware-evolution-2020/101029/

[13] Dar, M. A. (2017, March). A novel approach to restrict the access of malicious applications in android. In *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (pp. 1-4). IEEE.

[14] Investigation on Android Malware 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. (2019). Retrieved 18 August 2021, from https://www.unb.ca/cic/datasets/invesandmal2019.html

[15] Nguyen-Vu, L., Ahn, J., & Jung, S. (2019). Android fragmentation in malware detection. *Computers & Security*, *87*, 101573.

[16] Qin, Z., Xu, Y., Liang, B., Zhang, Q., & Huang, J. (2013). An Android malware static detection method. *Journal of Southeast University (Natural Science Edition)*, *43*, 7-1162.

[17] Xiong, P., Wang, X., Niu, W., Zhu, T., & Li, G. (2014). Android malware detection with contrasting permission patterns. *China Communications*, *11*(8), 1-14.

[18] Talha, K. A., Alper, D. I., & Aydin, C. (2015). APK Auditor: Permission-based Android malware detection system. *Digital Investigation*, *13*, 1-14.

[19] Fan, W., Liu, Y. A., & Tang, B. (2015). An api calls monitoring-based method for effectively detecting malicious repackaged applications. *International Journal of Security and Its Applications*, *9*(8), 221-230.

[20] Zou, S., Zhang, J., & Lin, X. (2015). An effective behavior-based Android malware detection system. *Security and Communication Networks*, *8*(12), 2079-2089.

[21] Wu, D. J., Mao, C. H., Wei, T. E., Lee, H. M., & Wu, K. P. (2012, August). Droidmat: Android malware detection through manifest and api calls tracing. In *2012 Seventh Asia Joint Conference on Information Security* (pp. 62-69). IEEE.

[22] Zhou, Y., & Jiang, X. (2012, May). Dissecting android malware: Characterization and evolution. In *2012 IEEE symposium on security and privacy* (pp. 95-109). IEEE.

[23] Zarni Aung, W. Z. (2013). Permission-based android malware detection. *International Journal of Scientific & Technology Research*, *2*(3), 228-234.

[24] Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., & Siemens, C. E. R. T. (2014, February). Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss* (Vol. 14, pp. 23-26).

[25] Jang, J. W., & Kim, H. K. (2016). Function-oriented mobile malware analysis as first aid. *Mobile Information Systems*, *2016*.

[26] Jang, J. W., Kang, H., Woo, J., Mohaisen, A., & Kim, H. K. (2016). Andro-Dumpsys: Anti-malware system based on the similarity of malware creator and malware centric information. *computers & security*, *58*, 125-138.

[27] Wang, Z., & Wu, F. (2015, April). Android malware analytic method based on improved multi-level signature matching. In *2015 5th International Conference on Information Science and Technology (ICIST)* (pp. 93-98). IEEE.

[28] Sharma, A., & Dash, S. K. (2014, October). Mining api calls and permissions for android malware detection. In *International Conference on Cryptology and Network Security* (pp. 191-205). Springer, Cham.

[29] Peiravian, N., & Zhu, X. (2013, November). Machine learning for android malware detection using permission and api calls. In *2013 IEEE 25th international conference on tools with artificial intelligence* (pp. 300-305). IEEE.

[30] Chan, P. P., & Song, W. K. (2014, July). Static detection of Android malware by using permissions and API calls. In *2014 International Conference on Machine Learning and Cybernetics* (Vol. 1, pp. 82-87). IEEE.

[31] Zeng, H., Ren, Y. A. N., Wang, Q. X., He, N. Q., & Ding, X. Y. (2014, December). Detecting malware and evaluating risk of app using Android permission-API system. In *2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing (ICCWAMTIP)* (pp. 440-443). IEEE.

[32] Ayşan, A. İ., & Şen, S. (2015, May). Api call and permission based mobile malware detection. In *2015 23nd Signal Processing and Communications Applications Conference (SIU)* (pp. 2400-2403). IEEE.

[33] Gascon, H., Yamaguchi, F., Arp, D., & Rieck, K. (2013, November). Structural detection of android malware using embedded call graphs. In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security* (pp. 45-54).

[34] Isohara, T., Takemori, K., & Kubota, A. (2011, December). Kernel-based behavior analysis for android malware detection. In *2011 seventh international conference on computational intelligence and security* (pp. 1011-1015). IEEE.

[35] Lin, Y. D., Lai, Y. C., Chen, C. H., & Tsai, H. C. (2013). Identifying android malicious repackaged applications by thread-grained system call sequences. *computers & security*, *39*, 340-350.

[36] Kiss, N., Lalande, J. F., Leslous, M., & Tong, V. V. T. (2016). Kharon dataset: Android malware under a microscope. In *The {LASER} Workshop: learning from authoritative security experiment results ({LASER} 2016)* (pp. 1-12).

[37] Lashkari, A. H., Kadir, A. F. A., Gonzalez, H., Mbah, K. F., & Ghorbani, A. A. (2017, August). Towards a network-based framework for android malware detection and characterization. In *2017 15th Annual conference on privacy, security and trust (PST)* (pp. 233-23309). IEEE.

[38] Liu, Y., Guo, K., Huang, X., Zhou, Z., & Zhang, Y. (2018). Detecting android malwares with high-efficient hybrid analyzing methods. *Mobile Information Systems*, *2018*.

[39] Lashkari, A. H., Kadir, A. F. A., Taheri, L., & Ghorbani, A. A. (2018, October). Toward developing a systematic approach to generate benchmark android malware datasets and classification. In *2018 International Carnahan Conference on Security Technology (ICCST)* (pp. 1-7). IEEE.

[40] Karbab, E. B., Debbabi, M., Derhab, A., & Mouheb, D. (2018). MalDozer: Automatic framework for android malware detection using deep learning. *Digital Investigation*, *24*, S48-S59.

[41] Taheri, L., Kadir, A. F. A., & Lashkari, A. H. (2019, October). Extensible android malware detection and family classification using network-flows and API-calls. In *2019 International Carnahan Conference on Security Technology (ICCST)* (pp. 1-8). IEEE.

[42] Ayodele, T. O. (2010). Types of machine learning algorithms. *New advances in machine learning*, *3*, 19-48.

[43] Newsom, I. (2015). Data Analysis II: Logistic Regression.

[44] Elder, J. (2015). Introduction to Machine Learning and Pattern Recognition. *Available at LASSONDE University EECS Department York website: http://www. eecs. yorku. ca/course_archive/2011-12/F/4404-5327/lectures/01% 20Introduction. pd*.

[45] Timothy Jason Shepard, P. J. (1998). Decision Fusion Using a Multi-Linear Classifier. In *Proceedings of the International Conference on Multisource-Multisensor Information Fusion*.

[46] Setiono, R., & Leow, W. K. (2000). FERNN: An algorithm for fast extraction of rules from neural networks. *Applied Intelligence*, *12*(1), 15-25.

[47] Sebestyen, G. (1966). Review of'Learning Machines'(Nilsson, Nils J.; 1965). *IEEE Transactions on Information Theory*, *12*(3), 407-407.

[48] Hormozi, H., Hormozi, E., & Nohooji, H. R. (2012). The classification of the applicable machine learning methods in robot manipulators. *International Journal of Machine Learning and Computing*, *2*(5), 560.

[49] Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, *160*(1), 3-24.

[50] Rakotomamonjy, A. (2003). Variable selection using SVM-based criteria. *Journal of machine learning research*, *3*(Mar), 1357-1370.

[51] Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

[52] Osisanwo, F. Y., Akinsola, J. E. T., Awodele, O., Hinmikaiye, J. O., Olakanmi, O., & Akinjobi, J. (2017). Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)*, *48*(3), 128-138.

[53] Hastie, T., Tibshirani, R., & Friedman, J. (2001). The elements of statistical learning. Springer series in statistics. In ∴ Springer.

[54] Tsai, C. F. (2009). Feature selection in bankruptcy prediction. *Knowledge-Based Systems*, *22*(2), 120-127.

[55] Ho, T. K. (1995, August). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition* (Vol. 1, pp. 278-282). IEEE.

[56] Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

[57] Khalid, S., Khalil, T., & Nasreen, S. (2014, August). A survey of feature selection and feature extraction techniques in machine learning. In *2014 science and information conference* (pp. 372-378). IEEE.

[58] Ladha, L., & Deepa, T. (2011). Feature selection methods and algorithms. *International journal on computer science and engineering*, *3*(5), 1787-1797.

[59] Rangarajan, L. (2010). Bi-level dimensionality reduction methods using feature selection and feature extraction. *International Journal of Computer Applications*, *4*(2), 33-38.

[60] Zheng, Y., Vanderbeek, B., Daniel, E., Stambolian, D., Maguire, M., Brainard, D., & Gee, J. (2013, April). An automated drusen detection system for classifying age-related macular degeneration with color fundus photographs. In *2013 IEEE 10th International Symposium on Biomedical Imaging* (pp. 1448-1451). IEEE.

[61] Ambusaidi, M. A., He, X., Nanda, P., & Tan, Z. (2016). Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE transactions on computers*, *65*(10), 2986-2998.

[62] Yang, Z., You, W., & Ji, G. (2011). Using partial least squares and support vector machines for bankruptcy prediction. *Expert Systems with Applications*, *38*(7), 8336-8342.

[63] Patil, A. M., Kolhe, S. R., & Patil, P. M. (2009, December). Face recognition by PCA technique. In *2009 Second International Conference on Emerging Trends in Engineering & Technology* (pp. 192-195). IEEE.

[64] Liang, D., Tsai, C. F., & Wu, H. T. (2015). The effect of feature selection on financial distress prediction. *Knowledge-Based Systems*, *73*, 289-297.

[65] Kadir, A. F. A., Stakhanova, N., & Ghorbani, A. A. (2015, November). Android botnets: What urls are telling us. In *International Conference on Network and System Security* (pp. 78-91). Springer, Cham.

[66] Gonzalez, H., Stakhanova, N., & Ghorbani, A. A. (2014, September). Droidkin: Lightweight detection of android apps similarity. In *International Conference on Security and Privacy in Communication Networks* (pp. 436-453). Springer, Cham.

[67] Total, V. (2012). Virustotal-free online virus, malware and url scanner. *Online: https://www. virustotal. com/en*, *2*.

[68] Total, V. (2016). Contagio mobile malware mini dump. *Online: http://contagiominidump. blogspot. ca*.

[69] Rakotomamonjy, A. (2003). Variable selection using SVM-based criteria. *Journal of machine learning research*, *3*(Mar), 1357-1370.

[70] Online banking users worldwide by region 2020 | Statista. (2021). Retrieved 18 August 2021, from https://www.statista.com/statistics/1228757/online-banking-users-worldwide/

[71] Mobile banking growth continues, 4-fold increase in 2015-16. (2021). Retrieved 18 August 2021, from https://www.cioandleader.com/article/2016/05/16/mobile-banking-growth-continues-4-fold-increase-2015-16

[72] Digital Payments in Pakistan (2021). Retrieved 18 August 2021, from https://portal.karandaaz.com.pk/dashboard/b873515a2a564150afa70a3413a99f38/1018