BEHAVIOUR ANALYSIS OF MOBILE SENSORS AND DETECTING UNUSUAL BEHAVIOUR

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

MASTER OF TECHNOLOGY

IN

SOFTWARE ENGINEERING

Submitted By:

SUKHDEV MATHUR

(2K18/SWE/20)

Under the supervision of

Dr. AKSHI KUMAR

DEPARTMENT OF COMPUTER SCIENCE

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

JUNE, 2020

DEPARTMENT OF COMPUTER SCIENCE

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, SUKHDEV MATHUR Roll No. 2K18/SWE/20 student of M.Tech Software Engineering, hereby declare that the project Dissertation titled "BEHAVIOUR ANALYSIS OFMOBILE SENSORS AND DETECTING UNUSUAL BEHAVIOUR" which is submitted by me to the Department of Computer Science, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi **SUKHDEV MATHUR**

Date:

DEPARTMENT OF COMPUTER SCIENCE

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled "BEHAVIOUR ANALYSIS OFMOBILE SENSORS AND DETECTING UNUSUAL BEHAVIOUR" which is submitted by Sukhdev Mathur, Roll No 2K18/SWE/20 Computer Science, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Akshi Kumar

Place: Delhi **Dr. AKSHI KUMAR**

Date: 24-09-2020 **SUPERVISOR**

DEPARTMENT OF COMPUTER SCIENCE

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I express my gratitude to my major project guide Dr. Akshi Kumar, Department of Computer Science, Delhi Technological University, for the valuable support and guidance she provided in making this major project. It is my pleasure to record my sincere thanks to my respected guide for his constructive criticism and insight without which the project would not have shaped as it has.

I humbly extend my words of gratitude to other faculty members of this department for providing their valuable help and time whenever it was required.

SUKHDEV MATHUR
Roll No. 2K18/SWE/20
Masters of Technology
(Software Engineering)

ABSTRACT

.

Today smartphones become a vital part of every individual in the world and smartphones have tied the users with a strong bond for every day to day task like setting alarm, order food, making payments and many more. A user never knows what going on inside their phones. A user cannot detect a mobile application whether it is having any malicious behaviour by its appearance, say you have downloaded an application from play store or any third-party store and that app is transmitting your personal data to a remote server without your knowledge. Even google play store sometimes cannot detect these applications due to code obfuscation techniques. In this research, we are analysing the behaviour of mobile sensors in malicious and benign mode and we are trying to detect if any application performs any malicious activity based on our analysis. We are using sherlock dataset for the behavioural analysis and applied four supervised machine learning techniques to detect unusual behaviour and comparing the results to find which technique is most accurate. We have taken 2 feature sets first contains only application features and other contain global features with application features. We have used F1 score as a deciding parameter for best performance. In results, we have found that XGBoost performs best with F1 score of 98.82 and 98.86 on applications global dataset respectively.

<u>**CONTENTS**</u>

LIST OF FIGURES

LIST OF TABLES

<u>**LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE**</u>

ABBREVIATIONS	**FULL FORM**
TPR	True Positive Rate
TNR	True Negative Rate
FPR	False Positive Rate
PPV	Positive Predictive Value
MSE	Root Mean Square Error

CHAPTER 1

INTRODUCTION

In 2020, there are 4.8 billion users out of which 3.5 billion users use smartphones worldwide [1]. Smartphones provide a way to communication, as well as a prime location for store and organizing information. As the number of applications increases for user's convenience, the amount of malicious applications harming users or breaching their privacy has also risen significantly. Currently there are 2.96 million apps on google play store and there are some other sources available for downloading the android applications. Many applications have acquired a place in our daily routine like an alarm application, messengers, email clients, gaming applications etc. A user never knew how an application operates in background. It is very difficult for a user to detect any malicious activity perform by an application; it is very possible that an application sends personal information of the user to a remote server without the consent of user.

In late 2012, a mobile malware was found on google play store named Android Dropdialer [3], this malware has self-updating capabilities. Applications infected with this malware that are present on google play store managed to bypass google play store security named bouncer because initially applications seem absolutely benign, had no malicious feature and malicious components were separately downloaded from internet known as remote payload technique. This feature made it possible for malware to stay on the market for long time and produce large number of downloads.

In 2017, another malware has managed to bypass google bouncer [4] known as Bankbot. This application has the capability to hide using code obfuscation time delay techniques. Many times, this application has taken down from google play store but every time this application found with its updated versions. This application was designed to steal user credentials from android devices, it is capable of bypassing two-factor authentication because it can monitor text messages. Smart phone users believe that applications on google play store are safe but their belief makes them more vulnerable to social engineering.

Generally, any malware can be installed on a device using remote payload technique. Malicious applications like Android Dropdialor and Bankbot uses remote payload technique. These types of applications are self-updating applications and cannot be detected using standard static techniques because original package of application does not contain any malicious component which appear as a benign application. Using dynamic code loading to transform a non-malicious application into a malicious application makes static analysis irrelevant. By using a time delayed or filtered deployment of the malicious payload dynamic analysis techniques get collapsed. For example, a hacker may provide friendly update on 1-5 days and on 6th day provide the malicious one. Moreover a experienced hacker can be code obfuscation techniques or code encryption to deter and complicate the dynamic or static analysis methods. These type of malware applications are difficult to detect because self-updating techniques are often used for benign purposes by lawful applications as well.

Many malwares are employed for stealing personal information, credentials or for ransom. So, our aim is to detect malicious applications which steals user information or spy on users. For this purpose, we have presented a method for detecting malicious applications using the behavioural analysis of applications i.e. how they are using the mobile resources and sensors. Proposed solution is intended to protect mobile device users from malicious applications that steal personal information and spies on users.

For conducting this research, we have used a rich dataset called Sherlock Dataset [11] provided by Ben-Gurion University. Dataset collected by two agents called Sherlock and Moriarty, there are 12 versions of Moriarty applications which are the replica of 12 different type of malwares, These Moriarty applications are known to be benign in conjunction with malicious behaviour. Moriarty change their behaviour malicious to benign and vice versa every few weeks. These application leave clues for the benign and malicious activities it performs and logged in raw Json format.

Our methodology is based on monitoring applications that run on a device and analysis the sensors behaviour. Any application that runs on a device consumes resources and operate on some sensors. For example, if a messenger application is running then it consumes some CPU usage, it will send and receive data from network sensors, it will consume some memory, battery and some other resources or sensors as per its requirements. For the purpose of anomaly in behaviour detection, we have proceeded the work in two steps. First, we analyse the behaviour pattern of these Moriarty applications in benign and malicious mode and in second, we use different models and train them on the dataset and find the best model who is able to predict the behaviour of application most accurately. We have shown the comparison of the classifier that we have used. We have used F1 score to select the best performing model. F1 score can be defined as the harmonic mean of precision and recall. We haven't used accuracy for selection of best performing model because accuracy is dependent on dataset we have used or on non-generalized factors. In the upcoming chapters we have presented background study, dataset exploration and analysis, behavioural analysis, model set up that we have used for our research and at last results and conclusion.

CHAPTER 2

BACKGROUND

2.1 MOBILE SECURITY THREATS

Usage mobile devices is on the ascent which lures the attackers to steal data from users. This put the devices at high risk of being attacked, It was found by the University of Cambridge that 87 percent of all android devices are vulnerable to at least one crucial flaw, while Zimperium Labs found not so long ago that with a straight forward instant SMS 95 percent of android devices could be hacked.

2.2 TYPES OF MALWARE

Any malicious application having some unusual behaviour like stealing data, lock the device for some ransom, showing ads, spying on user etc. On the basis of behaviour malwares are categorised as follows

Category	Description
Banking malwares	Collects the login and password details of social media applications, banking applications which are then sent to remote server without consent of users.
Mobile Ransomware	Ransomware "bolts out" critical user information, for example, documents, photographs and recordings by encrypting this data and afterward requesting a ransom to be paid to the malware producers. In the event that the ransom isn't paid on schedule asa rule in bitcoin all data are erased or essentially bolted up perpetually difficult to reach to the client.

Spyware	Spyware screens your actions, records your location and lifts basic data, for example, usernames and passwords for email accounts or web based business locales. By and large, spyware is bundled with other apparently benign applications and discreetly gathers information out of sight.
Adware	Adware shows irritating pop-ups and ads, now adware has made some amazing progress, attackers have now created malvertising code that can root and infect your device, constraining it to install different form of adware and facilitating attackers to access sensitive information.
Hostile Downloader	Initially benign, after installation it downloads the malicious components.

Table 2.1 Different types of malware

2.3 MACHINE LEARNING

Machine learning is the branch of artificial intelligence (AI) that gives systems the ability to automatically learn and improve on their experience without being directly programmed. Machine learning focuses on the development of computer programs that can access data and use it for learning. The learning process begins with observations or data, such as direct experience or teaching, to look at patterns in the data and make good decisions in future based on the examples we give. The primary goal is to allow computers to automatically learn without human assistance and to optimize operations accordingly. Machine learning can be divided into supervised learning and unsupervised learning.

In supervised learning, algorithm supposed to learn from the input variable (x) and produce the output variable (Y) and the mapping function from input to output is:

$$Y = f(X)$$

The goal is to better evaluate the mapping function, when you have new input data (x) you can estimate the output variable (Y) for that data.

In unsupervised Learning, we have only input data (X) and no relevant output variables. The goal of unsupervised learning is to model the underlying structure or distribution of data to learn more about data. unlike supervised learning above there is no correct answers.

Semi supervised learning is an approach in which model is trained on combination of and large amount of unlabelled data (Y) and small amount of labelled data (X).

Throughout this research, there has been a focus on supervised learning, we have tried to predict if there is any malicious application based on the behaviour of trained model with labelled data. The next section describes the classification of machine learning used in this research. The metric used to estimate the classification is then described in Section 2.4. Then section 2.5, explained detection methods by using machine learning and finally section 2.6 gives an insight of previous studies.

2.3.1 Logistic regression

Logistic regression is most appropriate for binary classification: datasets where y = 0 or 1, where 1 indicates the default class. [] For instance, in predicting whether an event will happen or not, there are just two prospects: that it happens (which we mean as 1) or that it doesn't (0). logistic regression uses logistic function

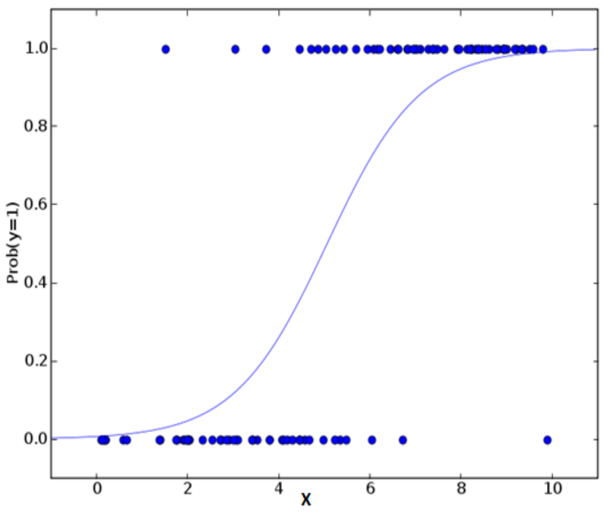

Figure 2.1 Logistic regression for binary classification

$$h(x) = \frac{1}{1 + ex}$$

so, it is called as logistic regression. this equation forms an S-shaped curve.

In logistic regression, the yield appears as probabilities of the default class. As it is a probability, the yield lies in the range of 0-1. it is exceptionally effective, doesn't require such a large number of computational resources, it's profoundly interpretable.

2.3.2 KNN (k- Nearest Neighbours)

Classification and regression problems can be solved by using KNN. It is a algorithm that stores all the possible cases and classifies new cases by majority votes of its k neighbours.

A case assigned to a class is most common among its k nearest neighbours measured by distance function. These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance. For continuous function first three functions are used and fourth one (Hamming) for categorical variables. If K = 1, then the case is simply assigned to the class of its nearest neighbour. At times, it turns out to be a challenge to choosing K while performing KNN modelling.

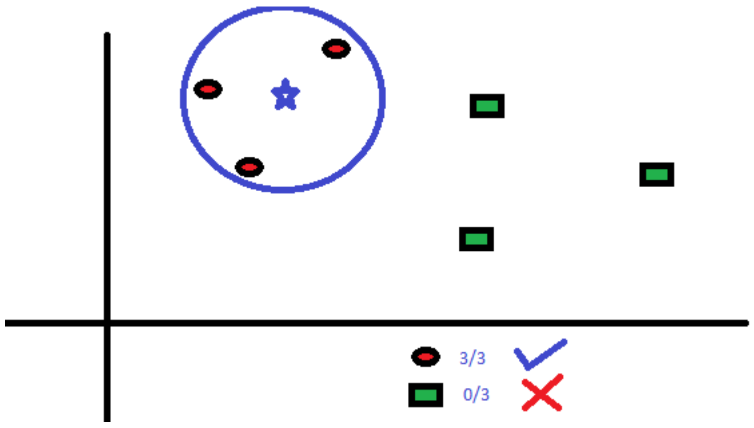

Figure 2.2 working of KNN

The algorithm is simple and implementation is easy. As tuning of parameter is not required, no need to develop a model or to make additional assumptions. But if dataset is too large then algorithm take larger time in training and testing i.e. algorithm becomes slow.

2.3.3 XGBoost

XGBoost is an implementation of Gradient Boosting Machines (GBM) and is used for supervised learning. XGBoost is an optimizd gradient boosting library intended to be exceptionally proficient, adaptable and versatile. Algorithm provides a parallel tree boosting (also known as GBDT, GBM) that solve huge problems with a fast and accurate method. It uses gradient descent in minimizing the loss function. The XGBoost has very strong predictive capabilities which makes it the best option for accuracy in events as it has both linear model and tree learning algorithm, rendering the algorithm nearly 10x faster than current gradient booster techniques. It has ability to handle thousands of input variables without deleting variable(s). Also, In the classification it can give estimates of variables which are important. But It take more time to train because of the fact trees are built sequentially and if the data is noisy XGBoost is more sensitive to overfitting.

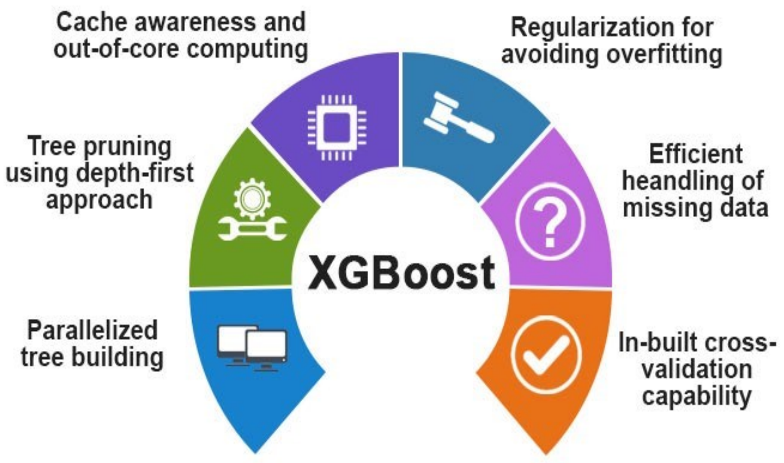
Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient heandling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

Figure 2.3 Description of XGBoost

2.3.4 Ensemble

In machine learning, we have a technique called Ensemble. It has the ability to combine multiple base models with the aim to create optimized predictive model. When making decision trees, rather of depending exclusively on one decision tree and assuming that we made the correct choice at each split, Ensemble methods helps one to take into consideration a set od decision trees, determine the features to use or ask questions at each split and create a final prediction dependent on the aggregated outcomes of the test decision trees. Ensemble has two types (i) bagging, it incorporates Bootstrapping and aggregation to create a single configuration ensemble. Multiple bootstrapped subsamples are pulled based on a sample of results. On each of the bootstrapped subsamples a

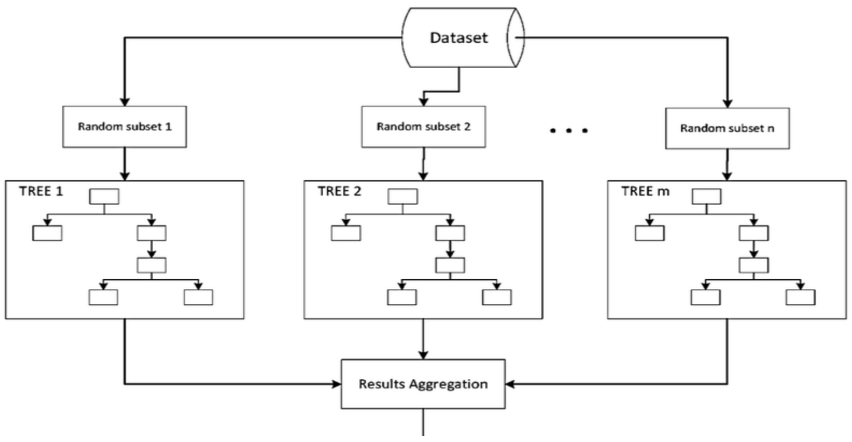

Figure 2.4 Bagging, most accurate prediction is choosen among all predictions

decision tree is created. After the creation of each sub-sample of decision tree an algorithm is used to sum across the decision trees to shape the most effective predictor.

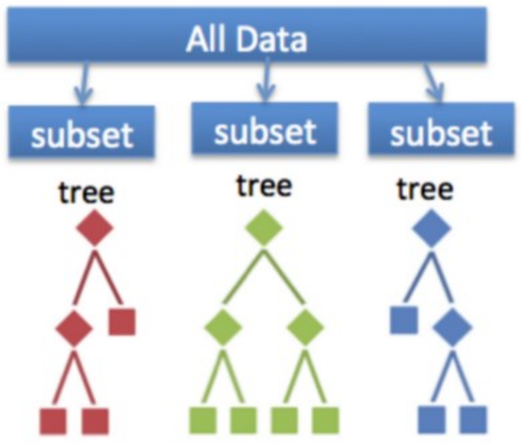

Figure 2.5 A random forest takes a random subset of features and create n random tree from each subset. Trees aggregated together at end

Another type is Random forest, we might think of random forest as bagging with a minor adjustment bagged decision trees have complete collection of options to pick from when deciding where to split and how to take decisions. Furthermore, while the bootstrapped samples vary significantly, the data will essentially break off at the same features in each model

On the contrary, Random Forest models determine where to split, depending on a random set feature. Rather of splitting at identical characteristics within each node, random forest models enforce a degree of separation as each tree can divide depending on different characteristics. This degree of distinction offers a larger whole for aggregating around, providing a more reliable indicator for ergo.

2.4 EVALUATING CLASSIFIERS

How can be measure the performance of a classifier? The obvious answer is to use Accuracy i.e. the number of given problems a model can classify correctly. There are many performance measures available to evaluate a classifier. The most basic performance metric is confusion metric [8]. A confusion matrix is as shown. Confusion matrix evaluated as:

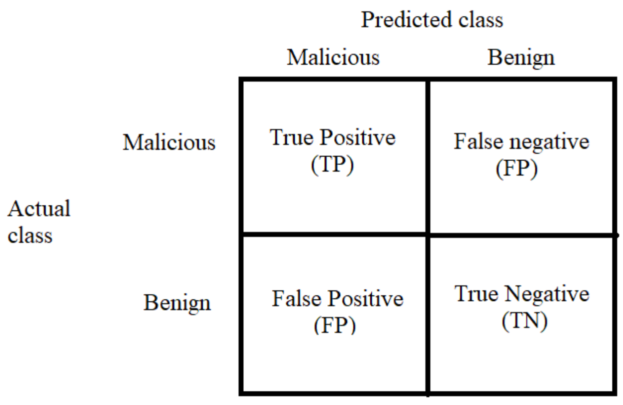

Figure 2.6 Confusion matrix

- If a malicious application is predictive as malicious then this is **True Positive**.
- If a benign application is predictive as benign then this is **True Negative**.
- If a benign application is predicted as malicious then this is a **False Positive**.
- If a malicious application is predicted as benign then this is a **False Negative**.

Accuracy can be defined as:

$$Accuracy = \frac{True\ positive + True\ negative}{True\ Positive\ + True\ Negative + False\ Positive + False\ Negative}$$

Metrics	**Formula**	
True Positive rate (TPR)	$\frac{TP}{TP+FN}$	Also known as Recall
False Positive rate (FPR)	$\frac{FP}{FP+TN}$	
True Negative rate (TNR)	$\frac{TN}{TN+FP}$	
Precision (PPV)	$\frac{TP}{TP+FP}$	
F-Score	$2 * \frac{Precision.Recall}{Precision+Recall}$	F-Score is defined as the harmonic mean of precision and recall.

Table 2.2 Performance Metrics

However, to measure the performance of a classifier accuracy might not be the best factor. In a skewed dataset there is higher records of first class than the other class. So by predicting the majority class high accuracy can be achieved. In such cases the performance metric Precision (PPV) or Recall (TPR) produce the more realistic results. The F1-Score shows the harmonic mean of recall and precision. Other Metrics are shown in Table 2.2

2.5 DETECTION APPROACHES

Below we have listed multiple techniques to detect mobile malwares which are as follows:

2.5.1 Static analysis

Static analysis is an easy and cheap way to detect malicious behaviour without running your app. The app is disassembled and either extracts system calls or creates a flow graph to detect if the app is malicious or harmless. The application is decompiled and the decompiled code is analysed. But this approach is limited to the number of new malware or malware variants [12].

2.5.2 Dynamic Analysis

In dynamic analysis, the applications are executed in sandbox or in an emulator to comprehend the dynamic behaviour like usage permission, system api call tracing, CPU usage, battery behaviour, RAM usage. Pollute following includes information stream investigation from sensitive sources like GPS, camera, Microphone and so on. In contrast to static investigation, it avoids the issues emerging from code obfuscation techniques and polymorphic behaviour [12].

2.5.3 Cloud based

The identification framework is enormously affected by the constrained processing power, restricted assets, limited battery. The analysis and identification some portion of calculation is moved to the cloud which has gigantic handling power. The monitoring part lies in the mobile environment. The monitoring part screens the Applications behave and procedures it before sending it to the cloud for location [12].

2.5.4 Anomaly based

In anomaly-based detection, the general behaviour of applications is compared to that of new applications. This approach can also detect new attacks or variants of known attacks with a high false alarm rate error. It uses machine learning algorithms that learn from trained data and determine whether the new test model is malicious or benign. It uses an unsupervised learning approach [12].

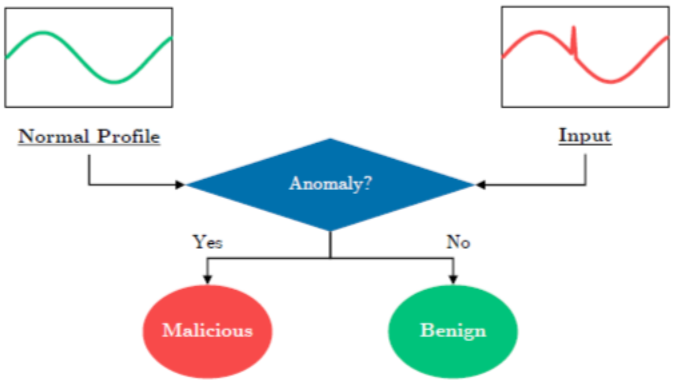

Figure 2.7 Anomaly based detection

2.5.5 Signature based

The behaviour of known attacks or malware is kept in the database as a signature model and the behaviour of new applications is checked against this database. Detection of this type can only be used for attacks which are known and has a low false alarm rate [12].

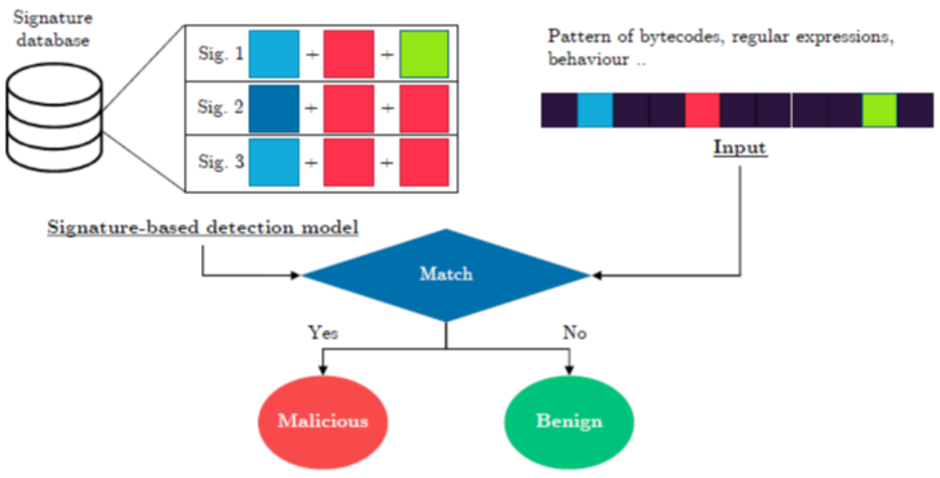
Signature database
Sig. 1
Sig. 2
Sig. 3
Pattern of bytecodes, regular expressions, behaviour ..
Input
Signature-based detection model
Match
Yes
No
Malicious
Benign

Figure 2.8 Signature based detection

2.6 RELATED WORK

For past many years many approaches are used to detect the malicious applications using different detection methods. Here we have listed some approaches. In 2011, shabtai et al published a paper in 2012 [13]. The authors of this paper have developed a framework named Andromaly with the purpose of detecting malicious applications using its behaviour. for this detection framework, they used total 88 features which were divided into 14 categories which are related to hardware, keyboard, touch screen, schedular. messaging, power consumption, memory usage, CPU load, applications, network, calls, processes, led and binder. Authors have utilized 4 self-developed malicious applications and 40 benign applications for data collection. These four malicious applications developed by them were a Spyware Trojan, DOS trojan, Spyware malware. and SMS Trojan. Authors have conducted four different experiments, on different devices with different models. Models were trained and tested on different sets of malicious and benign applications. The classifiers were used for train their model is Bayesian Network, J48, Histogram, K-means, Naïve Bayes and Logistic Regression. In both experiments, they have used same devices to train and test their model. Here J48 decision tree classifier has performed the best. For first experiment entire set of malicious and benign applications were taken in the ratio of 80% and 20% for training and testing. The experiment results in 99% of TPR and 0% of FPR. In second experiment, partial set of benign and malicious applications had taken. Training set were having 75% malicious, 75% benign applications and remaining set was used for testing which gives a TPR of 91% and an FPR of 11%. The Naïve Bayes classifier comes out as best performer in both the experiments. When complete set of benign and malicious applications had included in training set, as a result it leads to 91.3% of TPR and 14.7% of FPR. In another experiment where 75% of malicious applications used for the purpose of training and for testing remaining set was used from one device gives a TPR of 82.5% and an FPR of 17.8%.

Andromaly proved its capability of detecting malicious applications based on dynamic features using machine learning. Firmness of Andromaly also tested with changing the training devices from testing devices, by training and testing on different sets of training and testing. Yet the study is pretty outdated and since 2012 a lot has changed in terms of malware detection.

In [14], published in 2013, Author has proposed a method which is an anomaly-based detection method. This method uses behavioural features of an application. dataset used in this research contains features from 408 benign applications and 1330 malicious applications presented on google play store and database of malware genome project respectively. After evaluation of dataset, author found that network features and battery features were the same across dataset so only CPU, memory and Binder features were used. Another finding was that the benign features vectors were less in number compared to the malicious feature vectors so for balancing the dataset a technique was used called SMOTE. Naïve Bayes, Bayesian Network, Random Forest, Multilayer Perceptron, Logistic Regression, Decision Stump and J48 classifiers were used for research. Performance results were shown only for Random forest with different parameters. For the purpose of training and testing the model author used 5-fold cross validation method. The classifier which performs best had 8 different features, had 160 trees and had a depth of 16. These characteristics leads to root MSE of 0.0183%, accuracy of 99.9857%, and False positive were 2% only.

This study demonstrates the potential for using dynamic features and random forest classification. But in the real-world behaviour of model is not known.

In [15], published in 2013, For detection model, authors examined multiple machine learning techniques. Their model uses network, CPU usage, memory and SMS features. The author has used thirty benign and 5 malicious applications but source of these application is not mentioned. The malicious application set includes a Hostile Downloader, a Spyware, a Root[1,] Spyware and two trojan Spywares. Both malicious applications and benign applications were run and monitored under real environment however time duration of feature collection is not known. Dataset was too large so feature set size was reduced; the authors used an algorithm named information Gain algorithm. The reduced feature set contains features related to memory usage, Virtual memory, SMS and CPU usage. Logistic Regression, Naïve Bayes, SVM and Random Forest were used for evaluation. For a variety of malwares Random Forest shows excellent performance with TPR over 98.8% and FPR below 1%.

Research demonstrates the capability to use dynamic features but description of feature collection is insufficient which raise the question on reliability of performance.

In [16], published in 2014, For anomaly-based detection multiple hardware features were used, features related to battery, CPU, memory, ICMP requests and amount of connection requests. Data Collector is a data collector application which were installed in 12 smartphones for data collection. These smart phones have popular applications as benign ad three malicious applications. A Gaussian Mixture Model is used for detection with a Cluster-Based Local Outlier Factor which results in 100% TPR and 0% FPR.

Research demonstrate the ability of using user behavioural features with Gaussian Mixture Model for the detection of malicious applications. But reliability of the performance evaluation in hard to estimate because description of the feature collection is not given. Moreover, only three malicious applications were used in this research.

In [17], published in 2016, The authors had used memory usage and CPU usage features for dynamic detection of malicious applications. The model proposed in this research used a dataset of 1220 malicious applications from Malware Genome Project dataset and 952 benign applications from Google Play store. By running every application for 10 minutes separately in Android emulator Memory and CPU features were tracked. The emulator had run with Monkey application for user like inputs. The initial feature set consisted of 57 features and after data cleaning optimized dataset contained only 7 features. Logistic Regression algorithm was used with the use of sliding window technique. For training, authors had used 571 benign applications from Google Play store and 727 malicious applications from Genome project. There are 275 benign and 304 malware applications were used for testing. Finally, they used a validation set of 94 benign and 89 unseen malicious applications (previously means refers here that the malicious applications were neither in testing nor training set). The experiment results in TPR of 95.7% and FPR of 25% which is quite high. The detection model achieved TPR of 85.5% and FPR of 17.2% with highest F-score.

The research shows the capability of dynamic detection using memory usage and CPU usage for malicious application detection. But the model has shown the quite high FPR. Moreover, it is hard to estimate if the performance of model in real environment would be same or not as the authors had used android emulator with Monkey application to mimic the user behaviour.

In [18], published in 2016, In this research features used are related to CPU usage, system calls and Memory usage. Authors have found the three type of CPU features namely i) CPU user, ii) CPU kernel and iii) CPU total. Authors also observed three types of memory consumptions namely i) total memory usage, ii) native memory usage and iii) memory consumption by the Dalvik Virtual Machine. From per type of memory consumption, five features were taken out namely i) the private RAM, ii) the Heap allocation, iii) the Heap free, iv) the shared RAM and v) total proportional set size. A set of applications were made containing benign and malicious application where benign applications are from Google Play store and malicious applications are from Drebin dataset. The dataset was collected by running the set of applications in an Android emulator with Monkey application for 10 minutes and features were collected in every two seconds. Authors proceed with K-means clustering algorithm first to cluster applications based on similarity of CPU usage and memory usage. After that authors have applied Random Forest classification on applications cluster classified according to their system calls. to train their model, authors have used a set of 1000 benign applications and 1000 malicious applications. As results, the 7-means clustering and a Random Forest classifier of 50 trees had performed the best with FPR of 28% and an accuracy of 67%.

Research shows the ability of using Memory, CPU and system calls features for malicious application detection. But with the usage of an emulator with monkey application to mimic real user behaviour it is not known if the performance outcomes would be same in real environment or not. Moreover, this detection method uses System calls which requires root permission and performance is relatively low in comparison with other techniques.

In [19], published in 2016, authors use features related to Memory usage, CPU usage, Network usage and Storage for detecting malicious applications. A set of applications is prepared having benign applications from the Google Play store and malicious applications were taken from Drebin data. These Applications were run on android emulator with Monkeyrunner for 60 seconds. The authors have used Random Forest Classifiers with different type of parameters and Discrete cosine Transformation on features. Moreover, the authors amended the identification system using only global features, all features or only application features. The classifier uses a global features was comes out as a best performer, this model was trained with ten-fold cross-validations. The model led with 99.52% accuracy and FPR of 0.74%.

Research shows the ability to use Storage, memory usage, network and CPU usage features for the detection of malicious applications. An android simulator with the monkey application is used for monitoring behave of malicious applications separately that raise the question if the performance of model would be same in real environment or not.

In [20], published in 2017 A detection model had been proposed by authors using SVM which was based on CPU, memory and network usage of mobile phone devices. The features had system-wide and application specific monitoring. It is not known that how many malicious applications were there in the experiment as they had used drebin dataset but has not mentioned the amount. But drebin dataset has 5560 malware applications, the amount of malicious applications used is less than 5560. For the experiment, an android emulator was used with monkey application to mimic the real user environment. The C-SVM classifier has been used with a kernel for radial base function. the model was 82% accurate but the FPR is not mentioned and the precision ranges from 10% to 90% depending on the malware family.

By this research the Author has shown that his proposition has the ability to use Hardware features, but this model has done very poor, as FPR is much higher compared to other models. In addition, the authors have used the emulator and monkeyrunner to mimic a real user behavior that raises the question of how the model will work in the real world.

From the above researches it can be concluded that by using hardware features detection of malicious applications could be an effective way. It has been seen that K-Nearest Neighbour, Naïve Bayes, and Random Forest Classifier are most effective techniques. Many research papers have used emulators and the Monkey application to mimic the behaviour of real users in order to measure their performance on real devices. We also found that most researchers have run the malicious and benign applications for few minutes which limit the detection because if any malware shows the malicious behaviour after sometime it will not get detected. Moreover, limited research has shown high-performance outcomes.

Article		**Features**	**Performance**			
Ref	**Year**		**Classifier**	**Accuracy**	**TPR**	**FPR**
[13]	2012	Various(14)	BN, Histo, J48, Kmeans, LR, NB	0.809	0.786	0.475

[14]	2013	Binder, CPU, Memory,	RF, BN, NB, MLP, J48, DS, LR	1.000	-	0.02
[15]	2013	CPU, Network, Memory, SMS	NB, RF, LR. SVM	-	0.990	0.001
[16]	2014	Bat, CPU, Memory, Network	Gaussian mix-ture+LDCBOF	~1	~1	~0
[17]	2016	CPU, Memory	LR	-	0.855	0.172
[18]	2016	SC, CPU, Mem	Kmeans+RF	0.670	0.610	0.280
[19]	2016	Memory, Network, Storage, CPU	RF	0.995	0.820	0.007
[20]	2017	CPU, Memory, Network	C-SVM	0.820	-	-

Table 2.3: shows the different researches and their results

CHAPTER 3

DATASET EXPLORATION

In this chapter, we have portrayed the dataset used for this research. Section 3.1 explains dataset collection after that section 3.2 describes the available feature sets and features we are using for this research and last section describes our finding on the data.

3.1 DATA AGGREGATION

The data set [21] that we have used for the purpose of this research is provided by Cyber security center of Ben-Gurion University. For collection of datasets Mirskey et al. conducted data collection by providing Samsung S5 smart phones to 50 users and ask them to use these devices as their primary device. These smart phones were installed with self-written malicious applications which act like different malware types. The sherlock dataset collection starts from January 2015 and collected till December 2017, dataset for a year is divided into four quarters namely Q1, Q2, Q3 and Q4. One quarter contains the data for about four months. This dataset contains very precise information which show how smart phones are used. This data set contains usage data of different mobile sensors like cpu usage, battery usage, location related data, memory storage etc. Participants were instructed to use malicious applications at least ones in a day for few minutes. These is able to change their behave from benign to malicious and vice versa. Table 3.1 shows the self-written malicious applications and their description.

3.2 DATA DESCRIPTION

In [21], the features which are monitored are called as sensors, these sensors are group together so that at the same time they can be sampled together. These groups are named as probes, these probes are triggered in a fix time interval. Here sensors are divided into two categories i.e. PUSH and PULL where PUSH sensors are event based; it triggers when a button is tapped or any call arrives etc. PULL sensors collects data periodically like sampling the battery utilization or memory usage etc. There are 6 pull probes and 7 push probes. Table 3.2 shows the probes and description.

2016	Version	Application Behaviour		Description	Malware Type
		Benign	**Malicious**		
Q1	1	Puzzle game	Contacts Theft	Steals and transmits contacts	Spyware
	2	Web Browser	Spyware	Either, Location and audio spy Or, history spy and Web traffic	Spyware
	3	Utiliz. Widget	Photo Theft	Steals photos	Spyware
Q2	4	Sports App	SMS Bank Thief	Spies on SMS	Spyware
	5	Angry Bird	Phishing	Shows fake notifications to login in different apps like gmail, facebook etc.	Phishing
	6	Game	Adware	data gathering and shows popups, ads and banners.	Adware
Q3	7	Game	Madware	Information gathering, shows notifications and places shortcyts and triesto install new applications.	Spyware, Adware, Hostile Downloade r
	8	Lock Screen	Ransomeware	Locks the screen	Ransomwar e
	9	File Manager	Clickjacking	Useris tricked to trigger accessibility services and then it hijack the user interface.	Privilege escalation
Q4	10	None	Device Theft	Records the event when devices is stolen	
	11	Music Player	Botnet	Either SMS botnet activities or DDoS attacks	DOS

12	Web Media Player	Recon. Infiltration	Maps the connected local network and searches for files and vulnerabilities	Other

Table 3.1 Type of self-written application and description

Probe	No. of Field	Description
Call Log	5	Address, time, duration, outgoing or incoming, and an indication if number is from user’s contacts.
SMS Log	5	Address, time, outgoing or incoming, and an indication if number is from user’s contacts and if the content contains a URL.
Screen Status	2	Log of when the screen turns on or off.
User Presence	1	Android USER_PRESENT intent log: a record of when the user begins interacting with the device.
Broadcast Intents	3	All Android broadcast intents (events): changes in password, Bluetooth, network, RSSI, app packages, wallpaper, volume. Actions of button presses, picture/video taken, startup, shutdown, reboot, headset, phone ringing, notifications, TTS, and more.
App Packages	11	Log of when applications are installed, updated, or removed: provides the app’s version, hash of the APK, and list of permissions.
Moriarty	6	All clues left by the Moriarty malware agent.

Table 3.1 Description of the PUSH probes

Probes	Sample Interval	Sensors	No. of fields	Description
T0	1 day	Telephony Information	15	current telephony configuration information.
		Hardware Information	6	Device’s hardware configuration
		System Information	5	Kernel, SDK, baseband, and general information.

T1	1 minute	Location	15	{longitude, latitude, altitude, (anonymized via clustering)}, speed, and accuracy.
		Cell Tower	5	Cell tower ID, type and reception info
		Device Status	14	Brightness, volume levels, orientation and modes
		WiFi Scan	4	**For each visible AP**: identifiers, encryption, frequency, and signal strength.
		Bluetooth Scan	9	**For each visible device:** identifiers, device class (type), parameters, and signal strength.
T2	15 seconds	Accelerometer	51	Statistics on 800 samples captured over a duration of 4 seconds at 200Hz For each respective axis: mean, median, variance, covariance between axis, middle sample, FFT components and their statistics. A subset of these features is extracted from the orientation, rotation and barometer sensors.
		Linear Accelerometer	51	
		Gyroscope	51	
		Orientation	9	
		Rotation Vector	12	
		Magnetic Field	51	
		Barometer	16	
T3	10 seconds	Audio	21	Statistics over 5 seconds
		Light	3	Luminosity
T4	5 seconds	Global App stats	98	Information on the CPUs, memory, network traffic, IO interrupts, and connected WiFi AP.
		Battery	14	statistical data and configuration on temperature and power consumption.
Apps	5 seconds	Local App stats	70	For every running application: network traffic, CPU, memory.

	Linux level process data from the android system /proc folder

Table 3.2 Description of PULL probes

This research uses hardware sensor features for analysis of behaviour of malicious application and its detection, so we are using T4 probe, Apps probe and Moriarty probe for conceptual reference we have referred T4 probe as systems probe because it contains data for system. We have collected data for year 2016 for all four quarters which is of about 1TB but we are only using data of moriarty applications which is most relevant for our research and size of dataset reduced to about 9gb. In next subsection we have described about the probes which we are using for our research.

3.2.1 Moriarty Probe

Moriarty probe contains the data related to self-developed malicious applications described in Table 3.1. Each malicious application has the ability to change it behave from benign to malicious and vice versa in sometime we have identified three modes. In first mode, the session between benign and malicious for versions 1 and 11 are constantly changed. In second mode, the behaviour of version 2, 6 and 7 is changed after two malicious sessions to benign. In third mode, version 4, 5 and 8 are continuously remains in malicious session. Every malicious application has its behaviour exactly matched to a real malware as described in Table 3.1. Sherlock log every action and its details taken by the Moriarty applications. For user's data privacy and security sensitive information is encrypted. We have found that every record is mapped to a unique timestamp to track the action perform at a specific time. Moriarty performs solely benign actions in benign session, and in malicious session it can perform both benign and malicious actions. We excluded version 10 from this research because version 10 is for device theft simulation so it has no significance for this research.

3.2.2 System Probe (T4 probe)

System probes have global device data which is collected in every 5 seconds. The features logged are from CPU, battery, I/O interrupts, network, memory and storage there is also two columns which contains unique timestamp and userid. Each row contains global data for hardware sensors at a point of time for a user. Data of device had taken from /proc folder of android OS.

3.2.3 Apps probe

The Applications probe or apps probe contains data collected at every 5 seconds for every app that is installed on device. Each row contains data for user's application at a given time. There is also two columns one is unique timestamp and other is Userid for each record. For our research we have used data for Moriarty applications only. The app data is collected from /proc folder of android OS.

3.3 DATA SURVEY

To understand the contents of dataset, we have explored the all three probes individually and we find that ratio of malicious to benign records is 90:10. We have balanced the dataset by up sampling the data. Moreover, we also find that for Q1 and Q2 some data is columns are missing which are present in Q3 and Q4. We have omitted the columns which have constant values throughout the dataset and we have selected the columns having values related to CPU usage, memory usage, Network usage and battery consumption.

CHAPTER 4

DATA FEATURE ANALYSIS

This chapter has focused on data analysis and preparation for the modelling phase, section 4.1 describes the data selection procedure after that section 4.2 gives us the look towards the data cleaning methods, section 4.3 tells about the data integration phase and at last section 4.4 elaborate about the data balancing.

4.1 DATA SELECTION

For analysis of data, for training and testing of model data is selected related to CPU usage, network usage, memory usage. Data for Moriarty application is selected and data for other applications are dropped. Columns which are null, empty of having constant values are dropped. We have also selected UUID and userId for data integration but there is no need for these columns for data modelling.

4.2 DATA CLEANING

Imperfections of data are resolved so that it will not affect our final result. For overcoming data imperfections following steps are taken:

4.2.1 Removal of unwanted data

Unwanted data can be null values, duplicate values or irrelevant values. we have removed the columns which are entirely null and we convert the null values to 0 if exist in between of column. Duplicate values may be arise at the time of data collection. We have not found duplicate value before data integration but after integration there are duplicate values and we have removed them by selecting row on the basis of unique UUIDs. Irrelevant data is the data that does not fit for the solution of our problem and we have not found such data.

4.2.2 Resolving structural errors

Structural errors arise at the time of measurement, data transfer or other types of data handling activities. No such errors are found.

4.2.3 Filter unwanted outliers

Outliers are the observations that lie outside of the observation. Outliers distort and confuse the training cycle machine learning algorithm, leads to longer training times, less accurate models and worse performance. At the time of data analysis we find the unwanted outliers and we have successfully remove the outliers.

4.2.4 Handling missing data

Sometimes data is not present in some columns or rows, it is most tricky part to handle in data cleaning. To handle missing data, we have two ways one is to dropping the observations with missing values and other one is to fill the missing values from the past observations. In this dataset we have not found missing data. The features we have selected have not any missing data.

4.3 DATA INTEGRATION

System probe, Application prove and Moriarty probe are the three probes used through this research. For handling the dataset, we have used amazon Elastic Map Reduce cluster of 5 nodes with apache pyspark. As apache pyspark is capable of handling large amount of data, we have processed the data of one quarter at a time. All three probes are cleaned and features are selected from these probes and joined on the basis of *UserId* and *UUID*.

Data integration proceed with taking inner join between systems probe and application probe, we named it as *system_apps_probe* and after that this probe is merged with moriarty probe on *UserId* and *UUID* for one quarter and for other quarter we have repeat the same process and at last data of all quarter is joined to get a final clean dataset.

Merging of *system_apps_probes* is not so straight, as we have discussed earlier that records for applications probe and systems probes have collected in every 5 seconds so we have use the threshold of 5 seconds to join the Moriarty and system_apps_probes.

We have selected data of Moriarty application from applications probe because this data is most relevant to our research. Moriarty probes contains data from moriarty applications.

The final dataset is not so large so it can be fir into memory so the pyspark dataframe is converted into pandas dataframe and use pandas and python for modelling. We also have used the same dataset to analyse the behaviour of mobile sensors in benign and malicious modes using R.

4.4 BALANCING THE DATASET

As describes in section 3.3, The malicious data points are higher is number as compared to benign data points, they are in the ratio of 9:1. In real life malicious data points are lower than benign data points because malicious application work in a ay so that it can be hide among the benign behave. To balance the dataset we have two ways, these are i) to upsample the benign data points or downsample the malicious datapoints. We do not want any kind of data losses. So, we have upsample the benign datapoints. We have got the data points in the ratio of 50:50 for both benign and malicious.

CHAPTER 5

BEHAVIOURAL ANALYSIS

This chapter focus on the analysis of sensors which show how the sensors are working in benign and malicious mode. We have shown the analysis of cpu usage, bytes received and transmitted over the network and how memory is used by moriarty applications in malicious and benign modes.

5.1 CPU USAGE ANALYSIS

Every application that is running on your phone has some CPU utilization. For effective device performance CPU consumption by any application plays an crucial role. If an application consumes high CPU then it hampers the performance and slow down the phone. Moreover, if CPU consumption is high then it will also consume high battery. It is not necessary if an application is malicious then it will consume more CPU resources. It is worth noting that any benign application can have high CPU utilization say gaming applications but we need to portrait the behaviour pattern of CPU consumption for malicious and benign applications

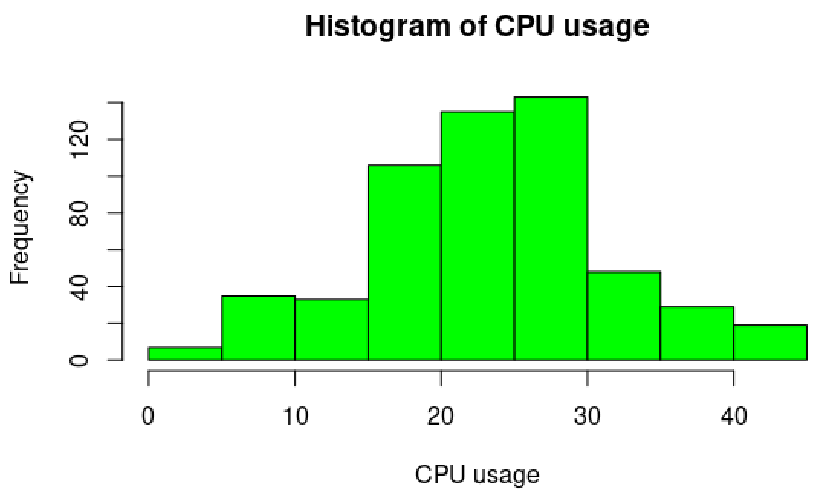

Figure 5.1 CPU usage behaviour

During the CPU usage analysis, we have found too high and too low values but for information we have clean the data, omitted the too low values and too high values were acting as outliers so we have adjusted them Figure 5.1 shows the CPU usage behaviour it can be clearly seen that CPU usage between 15 to 30 having very high frequency which tell in this interval CPU usage is very high. For interval 0 to 5 and 40 and above CPU usage is very low. Lastly For intervals 5 to 15 and 30 to 40 CPU usage is moderate, using the above information from the histogram we have plotted an Association plot in R to clearly visualize the CPU usage behaviour.

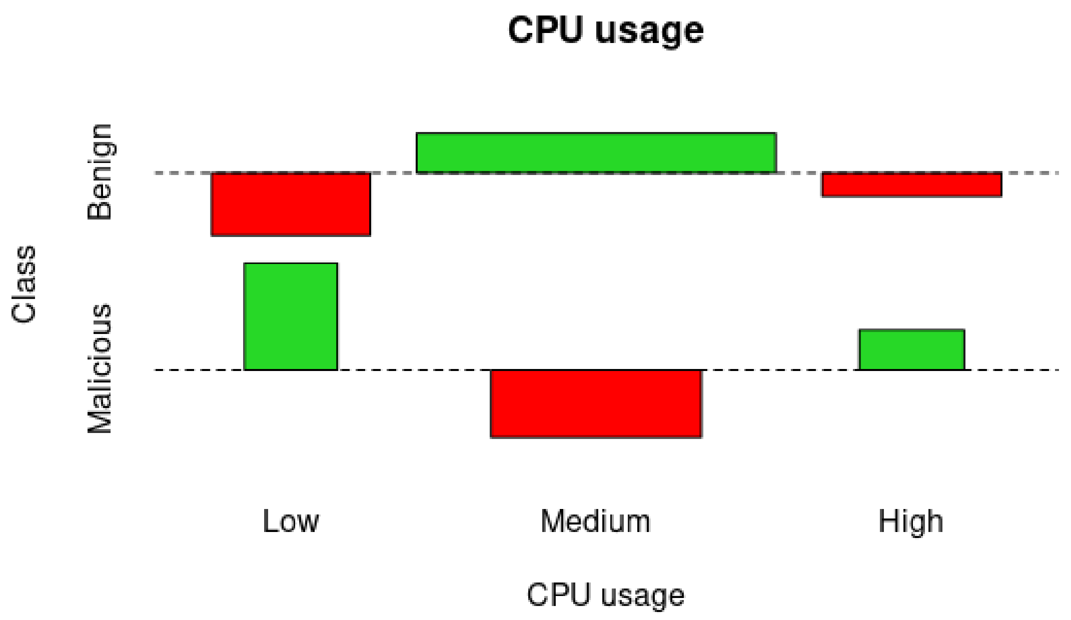

Figure 5.2 Association plot for CPU usage behaviour

An Association plot is a good choice to plot the proportion of observations for different categorical variables. Assocplot has a set of bar charts, showing the deviation of each combination of factors from independence. In figure 5.2, shows an Association plot between the CPU usage and benign and malicious behaviour. Here CPU usage is categorised in Low, Medium and High. It is clearly visible from the plot that if the CPU utilization is low or high then the behaviour is malicious and if the CPU usage is moderate then the behaviour is benign. In conclusion, We have found that if CPU malicious applications have too high or too low memory usage.

5.2 NETWORK USAGE ANALYSIS

In this section, we have focused on the received and transmitted bytes over network of android device. We have taken the total bytes received and transmitted over Wi-Fi or mobile data. In section 5.2.1 the behaviour of bytes transmission is described and section 5.2.2 describes the behaviour of bytes received.

5.2.1 Bytes transmitted analysis

In this section, we have highlighted the behave of bytes transmitted over the network. Today almost every application interacts with network and has some data transmission to remote server. For example, whatsapp, Ludo etc. These apps continuously sync with the server to prevent the data loss. In our research, Moriarty applications show both benign and malicious behave some of them are spywares which transmits the data to remote server but this data is in encrypted form to prevent user privacy. We have plot a histogram to show the bytes transmitted over the network.

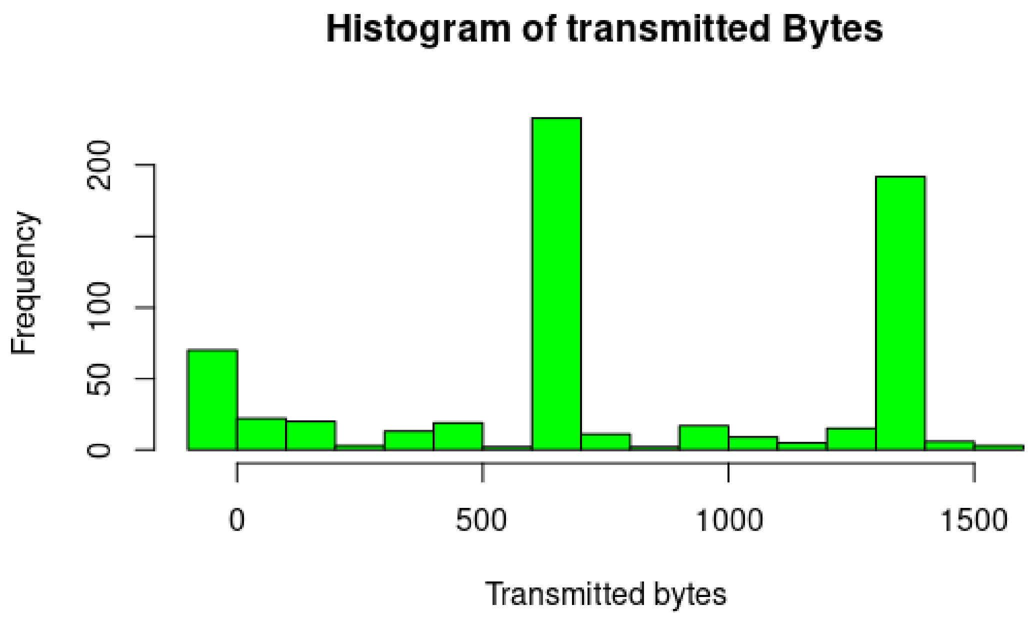

Figure 5.3 Histogram for bytes transmission analysis

In figure 5.3, it is clearly shown that bytes transmitted frequency is either very high or very low, for intervals 600 to 700 and 1300 to 1400 bytes transmission frequency is very high and for other intervals bytes transmission frequency is very low. For clear visualization of bytes transmission, we have plotted an Association graph for which shows the behaviour between the categorical variables that is benign and malicious.

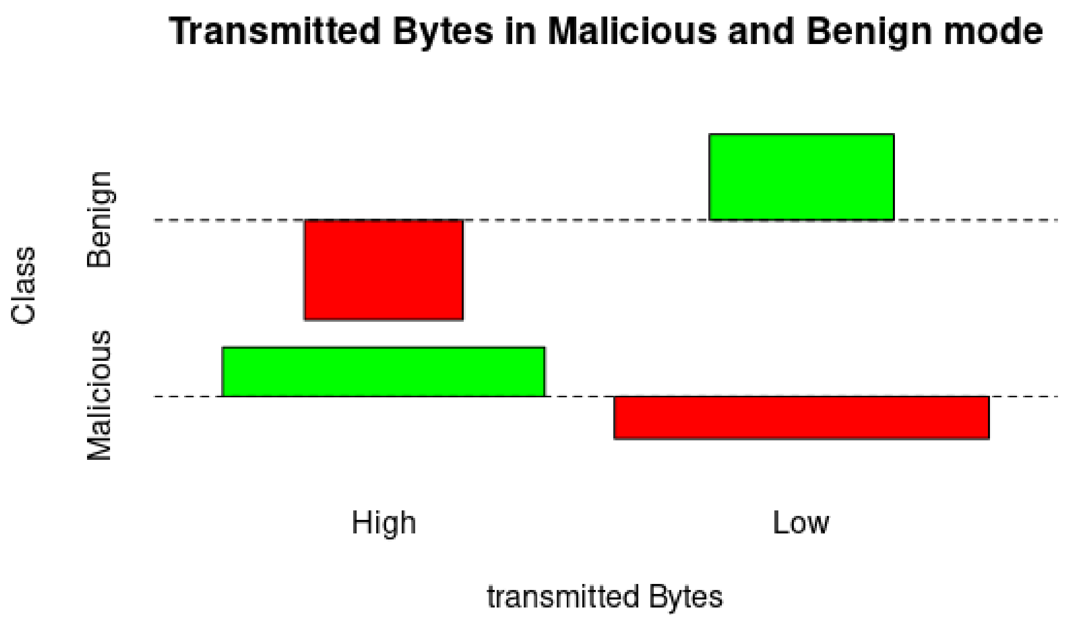

Figure 5.4 Association graph for bytes transmission analysis

Figure 5.4 shows the categorical distribution of bytes transmission, it is clearly seen that in malicious mode the transmission frequency of bytes is high rather than in benign mode the transmission frequency is low. In conclusion, we can say that malicious application tends to transmit higher data than the benign applications. A spyware has its prime goal of data collection from user device and send it to remote server whereas any benign applications have lower frequency of data transmission.

5.2.2 Bytes received analysis

Every byte transmitted over the network have got acknowledgement in return to ensure that data packet has been received successfully. We also try to analyse the received bytes in both benign and malicious modes. We have plotted a histogram to see the byte reception over the network on a android device.

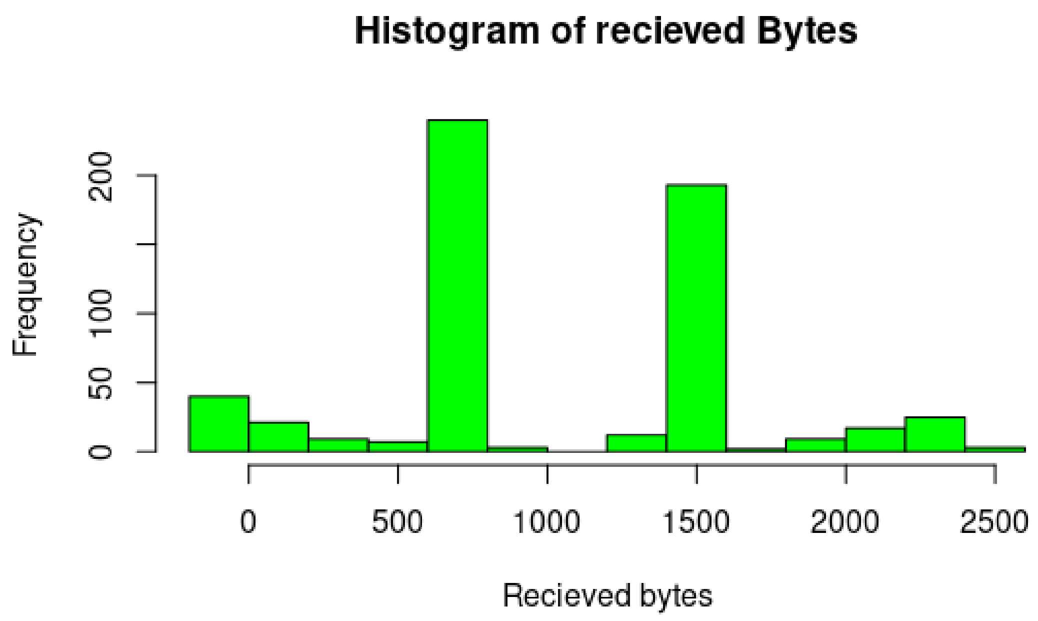

Figure 5.5 Histogram shows the bytes received over the network

Figure 5.5 shows that the frequency of receiving bytes over the network is very high in 600 to 800 and 1400 to 1600 and for other intervals the frequency is very low. Using the information from above histogram we have plotted a Association graph.

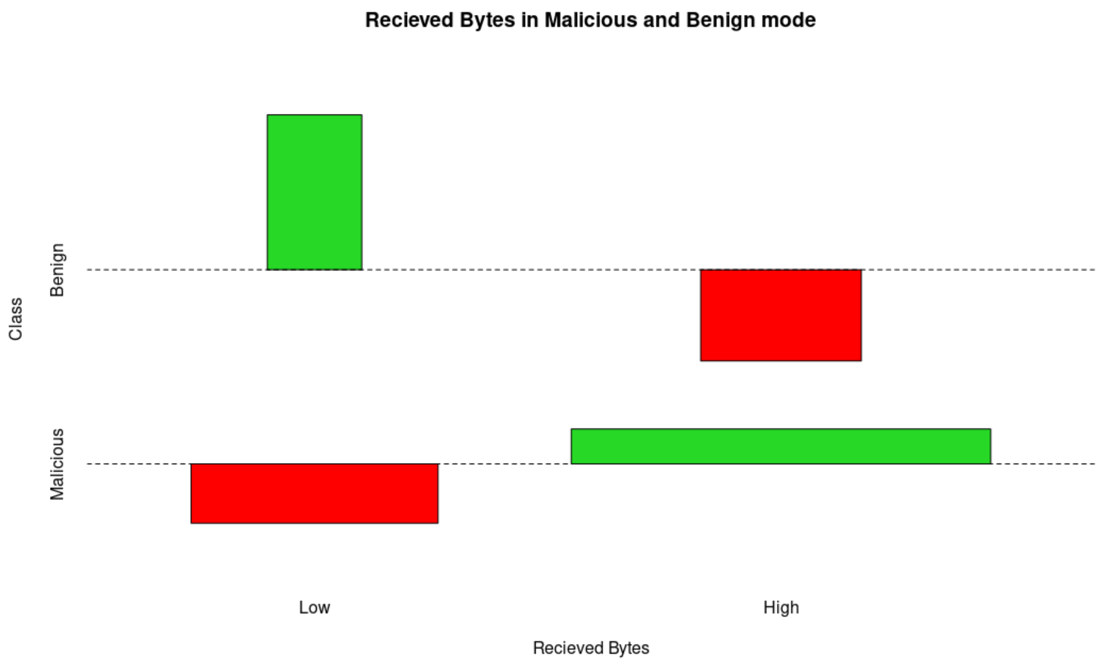

Figure 5.6 Association plot for bytes received over the network

In figure 5.6, It can be clearly seen that in benign mode receiving frequency of bytes in benign mode is low and in malicious mode frequency is very high.

Here, we have found that in malicious mode data transmission over the network is high as compared to benign mode, the fact behind this is malicious application tends to transfer data to a remote server without the consent of user. Moreover, frequency of data transmission is not so frequent, in figure 5.4 and figure 5.2, data transmission frequency is high between two intervals. In conclusion we can say that malicious applications send data but not in a frequent manner.

5.3 MEMORY USAGE ANALYSIS

Every application that is running on a mobile device has some memory consumption that is, it requires RAM for execution. If there are many running application in the background then it overloads the RAM. Every process that is run by any application is associated with a CPU thread and CPU requires data for processing, this data is accumulated in RAM from ROM. When process completes its execution CPU threads are released and RAM become free. For a malicious application it is not necessary that it will overloads the RAM or CPU usage but it may be possible that malicious application try to stop execution for other process and force the CPU to act maliciously which can lead to memory overheads.

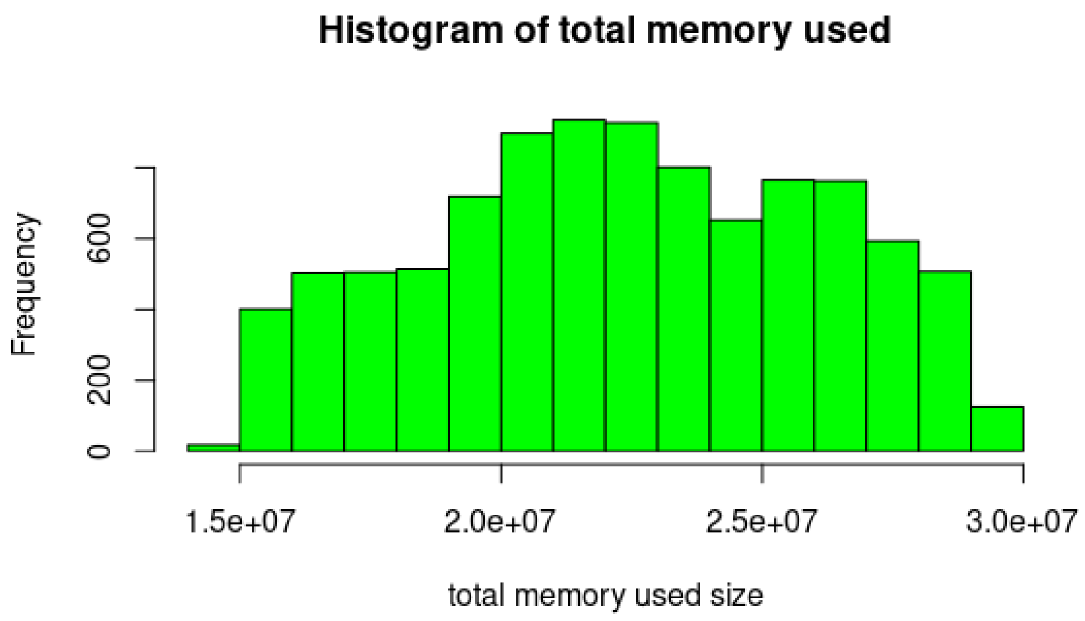

Figure 5.7 Memory usage analysis

Figure 5.7, shows the memory usage pattern, for this histogram we divide the bars in two categories that is high and low. For the intervals where memory usage has frequency less than 500, we have put them in low and above 500 we have put them in high and we plot an association plot for deep analysis.

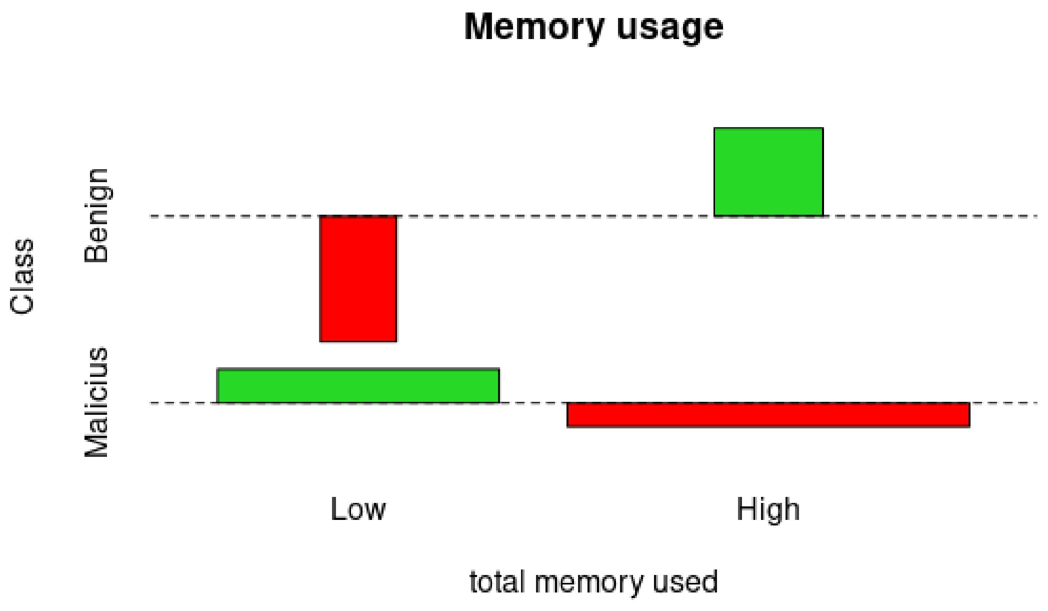

Figure 5.8 Association plot for memory usage

From Association plot in figure 5.8, it can be clearly deduced that in malicious mode memory usage is low and in benign mode memory usage is high. So, we can conclude that Moriarty applications that are acting as a malicious application does no effect memory.

In this chapter, we have seen the analysis of CPU usage, network and memory and how malicious and benign applications effect device sensors. In next chapter we will see the modelling algorithms that we have used to train and test our model.

CHAPTER 6

MODELLING

In this chapter, we have focused on modelling phase of this research. We have described how the dataset is used for the purpose of training and testing of the classification model. Selection of machine learning algorithms is described in first section. Section second describes the setup used for the research. Section third describes the training and testing of machine learning classifiers.

6.1 SELECTION OF MACHINE LEARNING ALGORITHMS

On the basis of analysis of previous studies, we have selected logistic regression, KNN, XGBoost and random forest. In our we have found that these algorithms have high TPR rate and low FPR and FNR and these algorithms have shown promising results for binary classification. Therefore, we have used above specified algorithms.

6.2 EXPERIMENT SETUP

6.2.1 Label

We are using binary classification for this research as we are classifying the application in Benign or malicious. Benign is indicated by 0 and malicious is indicated by 1. So Benign and malicious can be refer as labels that a classifier tries to predict. In moriarty file there are two fields called *ActionType* and *SessionType. ActionType* indicates that if a action taken by application is benign or malicious and *SessionType* indicates the current session under which an application is running as describes in section 3.2.1. As classifier requires numerical values to perform any operation so we have transformed the string values to numerical values where Benign is 0 and Malicious is 1.

We have labelled each row of the dataset as malicious or benign where *ActionType* and *SessionType* both are malicious as malicious else benign.

6.2.2 Feature set

The set of features that we have used for training and testing of our classifier referred as feature set. Our feature set includes Battery features, CPU usage features, network transmission features and memory usage features. These features are from global usage and application specific usage.

We have removed all metadata features which are *UUID, UserId, Version* from final merged dataset. We have appended a column which is a label field called label. We have assigned the label malicious if *ActionType* and *SessionType* both are malicious (1) else we label other rows as benign (0). We have then removed the *ActionType* and *SessionType* columns also as they serve no purpose after labelling the dataset. Final labelled dataset is in ratio of 10:90 w.r.t benign and malicious, so we have up-sampled the dataset to balance the dataset as described in section 4.4.

We have used two datasets for training and testing first comprises application features only and second comprises global features with application features. We have train and test model for both the models and compare their results.

6.2.3 Technologies

For this research, we have used AWS Elastic Map Reduce with Apache Pyspark with 5 node cluster with 32GB of RAM and 8 cores of CPU. Apache Spark use Hadoop for file management and Hadoop is built over Yarn and HDFS which give the capability to Hadoop to store, replicate and retrieve large amount of data. For data processing we have used pyspark as it can process data among multiple nodes. To train and test the model Scikit-learn is used which is a machine learning classifier package. Lastly, we have used zeppelin note book for code the classifier as it is provided by EMR. We have extracted the usable clean dataset for model training and testing which is smaller in amount so we have processed the model on the local system only to reduce the cost of AWS EMR. Locally we have used jupyter and same libraries for classification. Jupyter and zeppelin are almost same but both are good in their own way. For Data analysis we have used R.

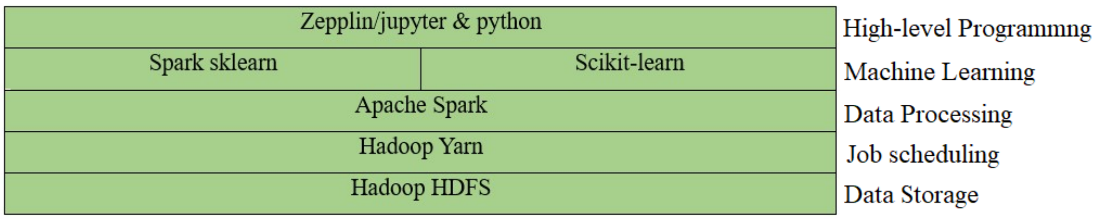

Figure 6.1 Data Analysis Tool structure

6.3 TRAINING AND TESTING

For training and testing and used the setup we have described in previous section. We have split the data in the ratio of 80:20 for training and testing. For testing we have removed label field. We have used F1 score for the selection of best model. F1-Score is the harmonic mean of the precision and recall. We have set the metrics of different classifiers in the table for final conclusion.

CHAPTER 7

RESULTS AND DISCUSSIONS

In this chapter, results of experiment have shown. We have described the result for every classifier one by one. In first section we have described the result of KNN then in section second, we have described the results of logistic regression. In section third we have described XGBoost, in fourth section we have ensemble. In fourth section, we have state the final verdict and finally we have described the usability and limitations.

7.1 KNN

The performance of KNN with both datasets has shown below. Confusion matrix for KNN is shown in figure 7.1 for applications features only and the normalized confusion matrix shown in figure 7.2.

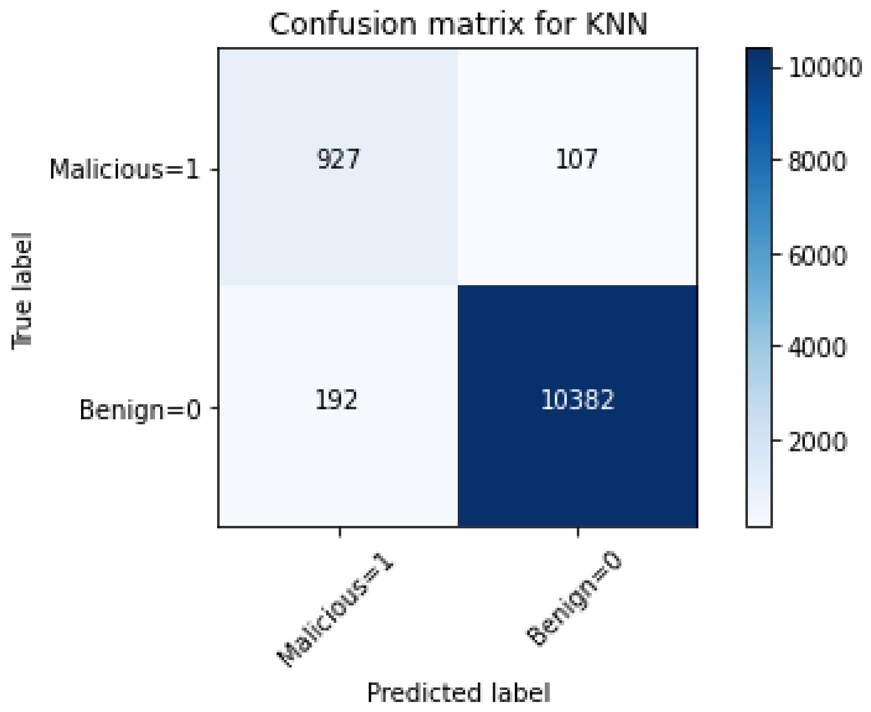

Figure 7.1 KNN confusion matrix using applications features

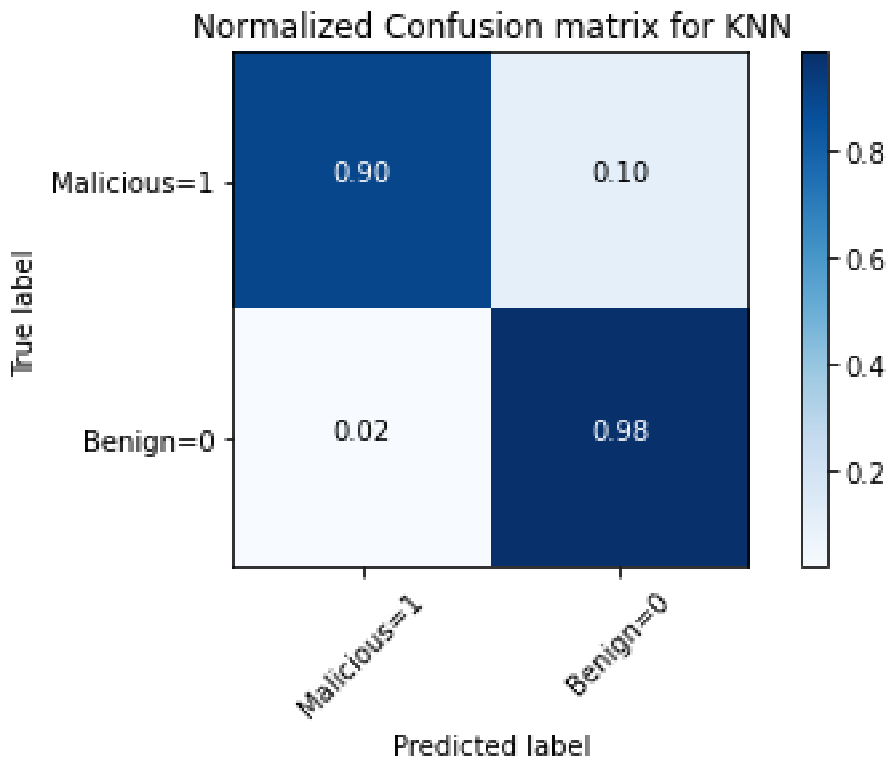

Figure 7.2 KNN normalized confusion matrix using applications features

For applications featureset, we can see from figure 7.1 and figure 7.2 that True positive is 90%, i.e. 90% of the malicious applications are classified as malicious and True negative is 10% which tells that 10% of malicious applications are miss classified. For benign applications False negative is 98% i.e., 98% of benign applications are predicted as benign and 2% of benign applications are miss classified as shown by false positive.

For global featureset, we can see from figure 7.3 and figure 7.4 that True positive is 72% i.e., 72% malicious applications are predicted correctly and 28% of malicious applications are miss classified as True negative is 28%. Whereas False negative is 96% means 96% benign applications are classified correctly and 4% are miss classified as show by False positive is 4%.

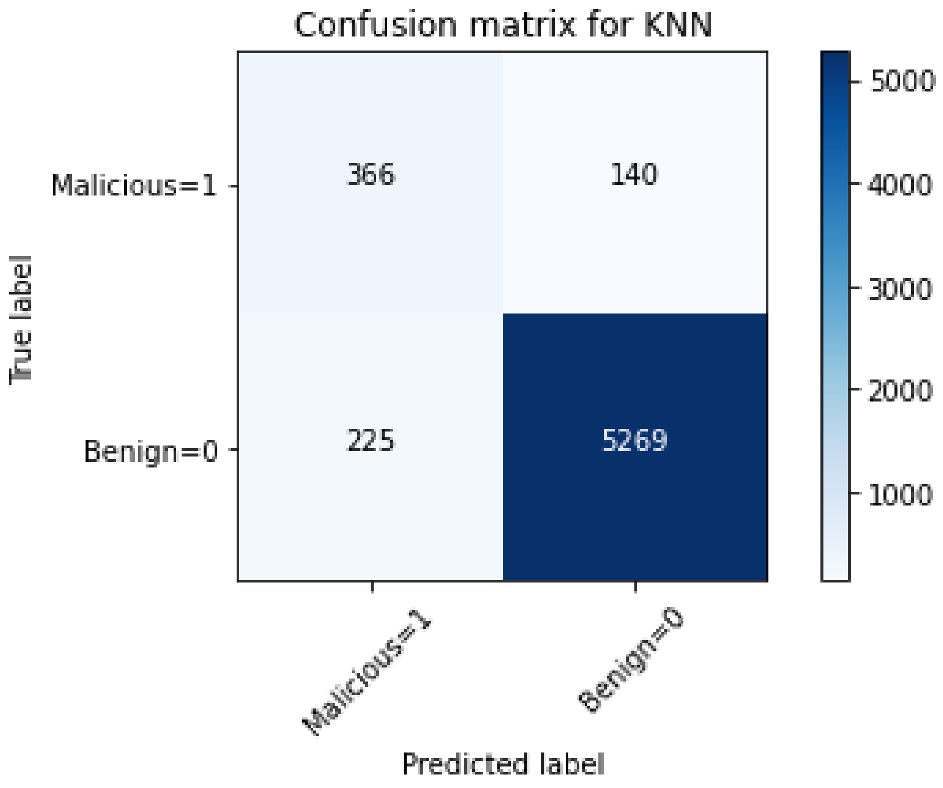

Figure 7.3 KNN confusion matrix using global features

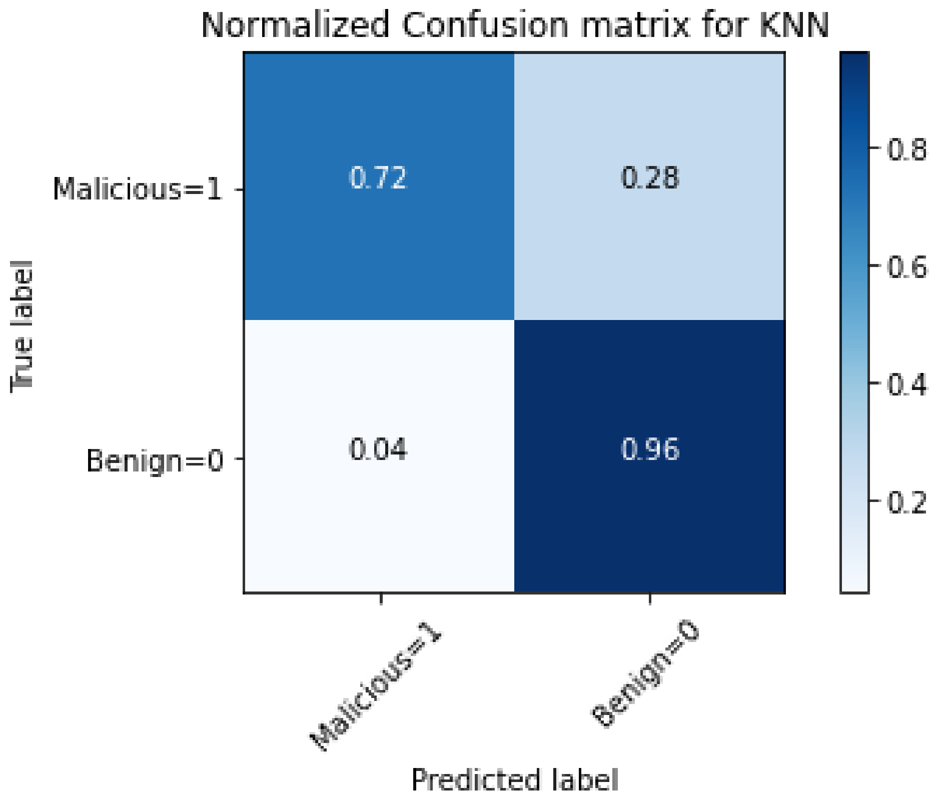

Figure 7.4 KNN normalized confusion matrix for global features

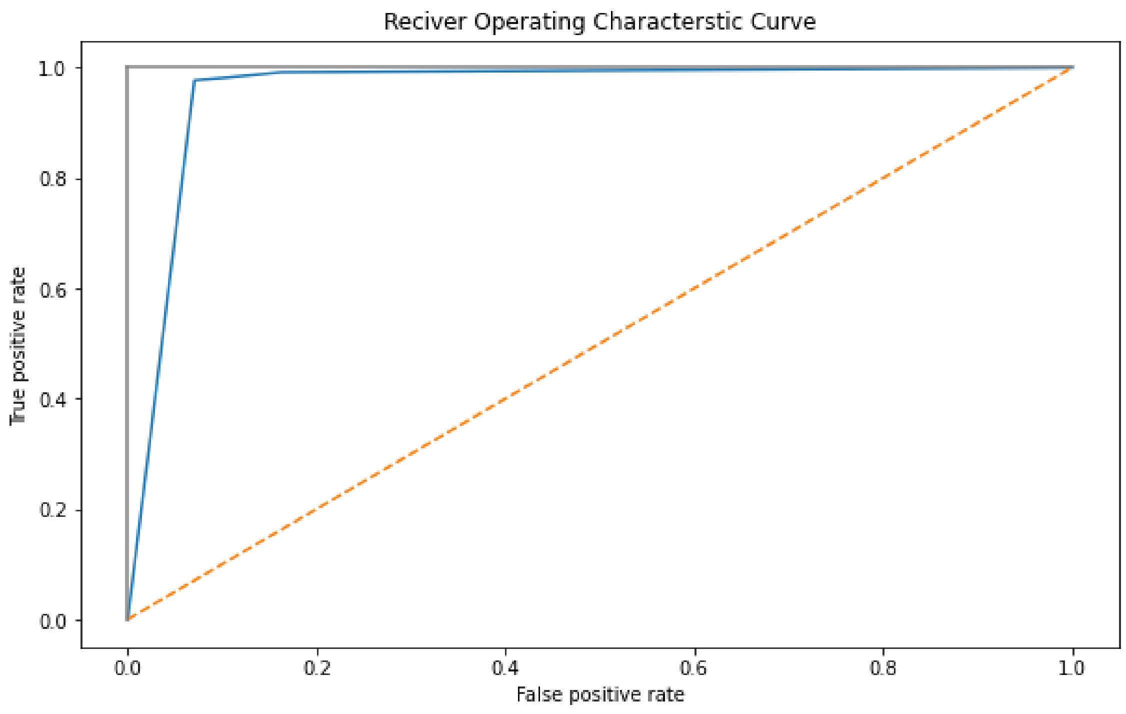

Figure 7.5 KNN ROC AUC using application features

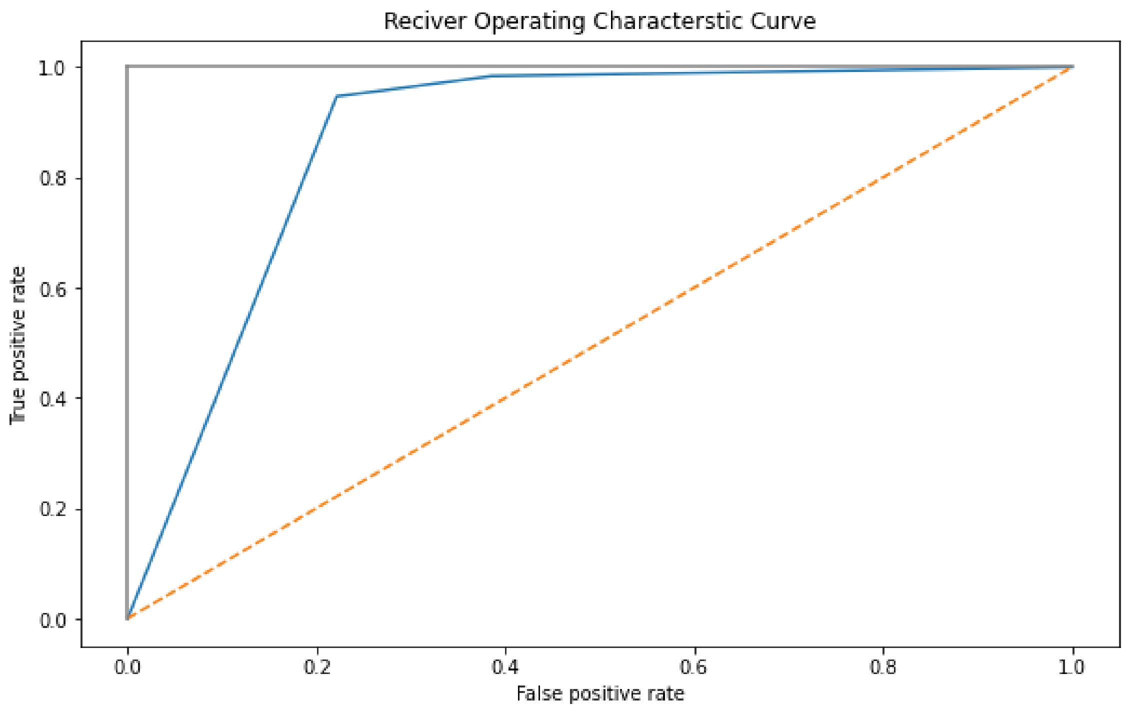

Figure 7.6 KNN ROC AUC using global features

Figure 7.5 and figure 7.6 show the AUC ROC curve for application feature set and global feature set respectively. We have found ROC for application feature set is

95.89% and for global feature set is 87.27% respectively. From the figure and ROC values we have find that application feature set are more promising for malicious application detection whereas global feature introduces noise in data and reduce the prediction ability for the model. Table 7.1 gives a result over view for the KNN.

KNN		
Featureset	Application features	Global features
Accuracy	97%	94%
F1 score	98.58%	96.65%
Recall	98.18%	95.90%
Precision	98.97%	97.41%
AUC	95.89%	87.27%

Table 7.1 Results for KNN

From Table 7.1, we can clearly see that Application feature set has greater accuracy, F1 score, Recall and precision which shows that we will find good results if we use applications feature set with KNN for detection.

7.2 LOGISTIC REGRESSION

The performance of logistic regression with both the feature set is described in this section. Figure 7.7 and figure 7.8 show the confusion matrix and normalized confusion matrix of applications feature set for logistic regression. We can clearly conclude from confusion matrix that True positive is 83% and False negative is 84% which means 83% of malicious applications predicted as malicious and 84% of benign applications predicted as benign where 17% is True negative and 16% is False positive which tells 17% malicious applications and 16% benign applications are misclassified.

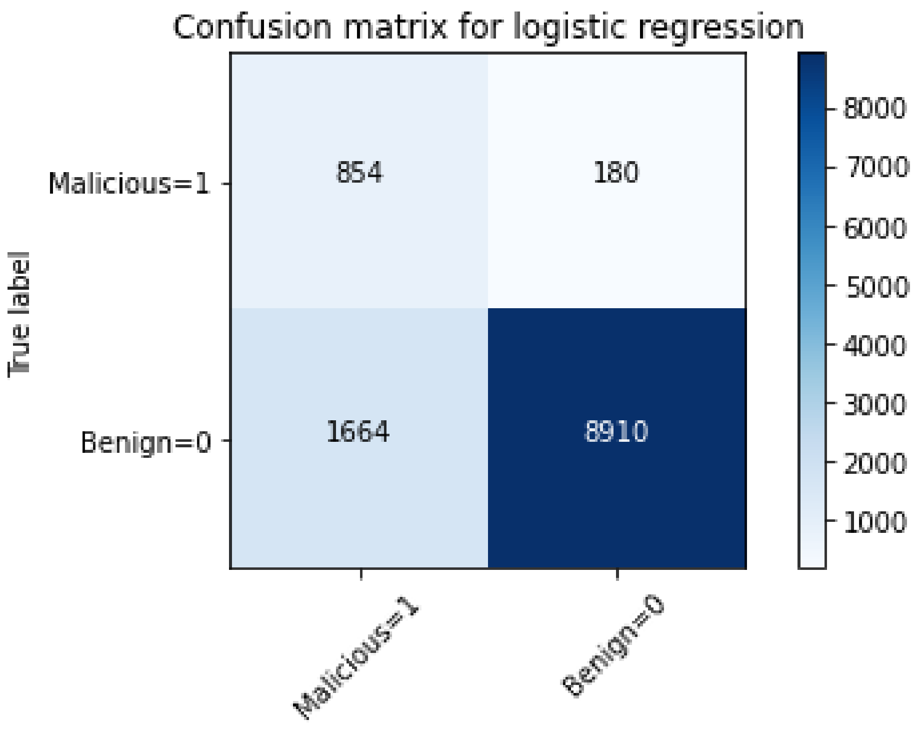

Figure 7.7 Logistic regression confusion matrix for applications features

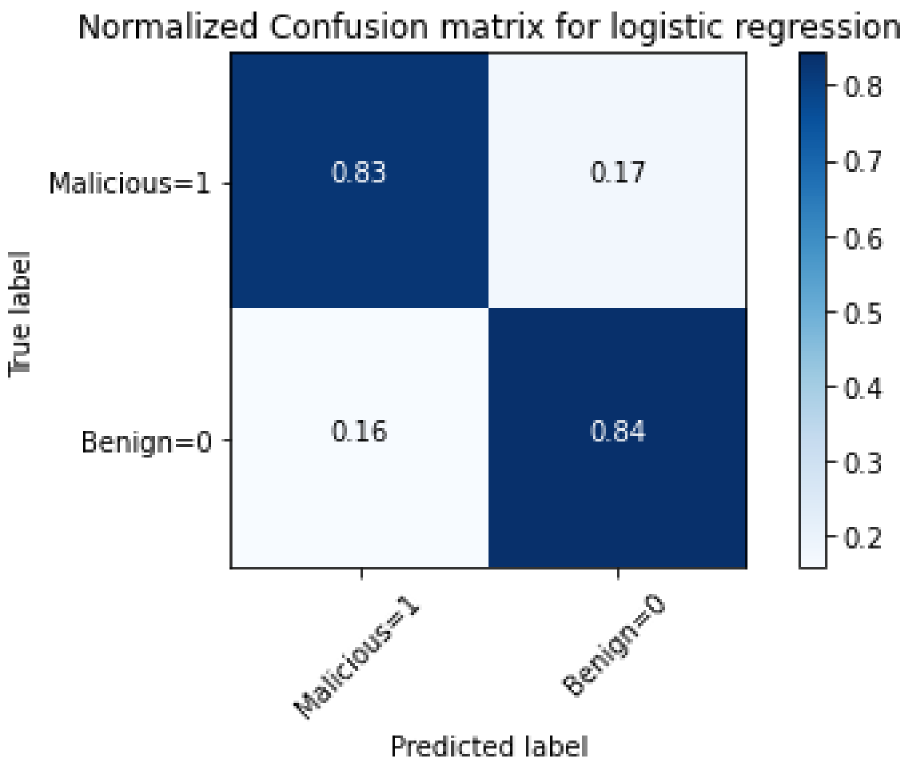

Figure 7.8 Logistic regression normalized confusion matrix for applications features

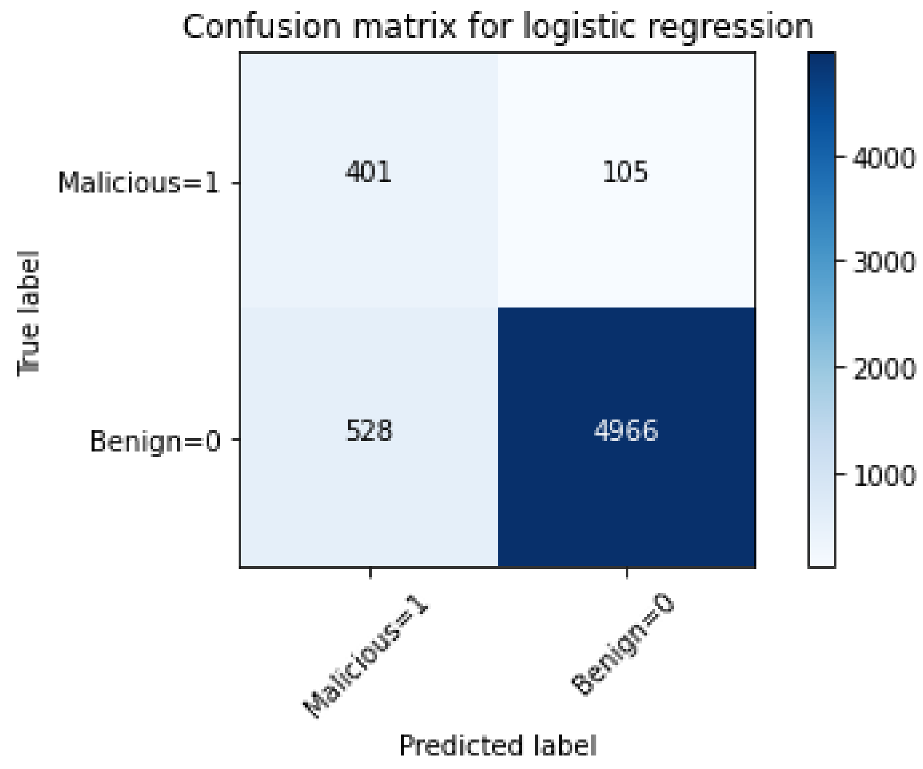

Figure 7.9 Logistic regression confusion matrix for global feature set

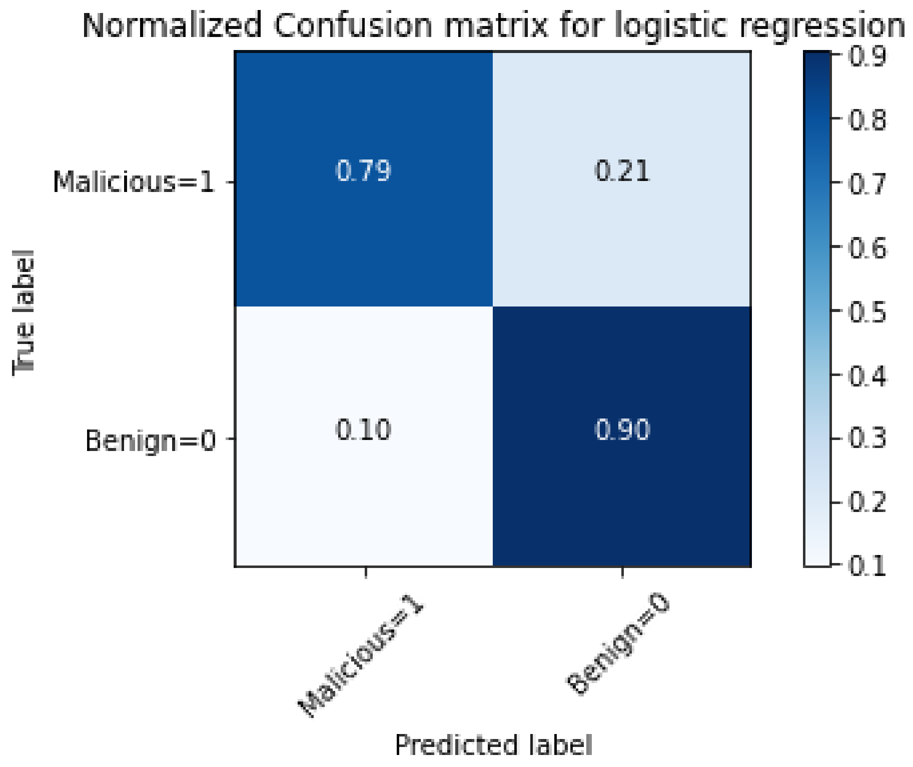

Figure 7.10 Logistic regression normalized confusion matrix for global feature set

Figure 7.9 and figure 7.10 for global feature set, we can clearly conclude from confusion matrix that True positive is 79% and False negative is 90% which means 79% of malicious applications predicted as malicious and 90% of benign applications predicted as benign where 21% is True negative and 10% is False positive which tells 10% malicious applications and 10% benign applications are misclassified.

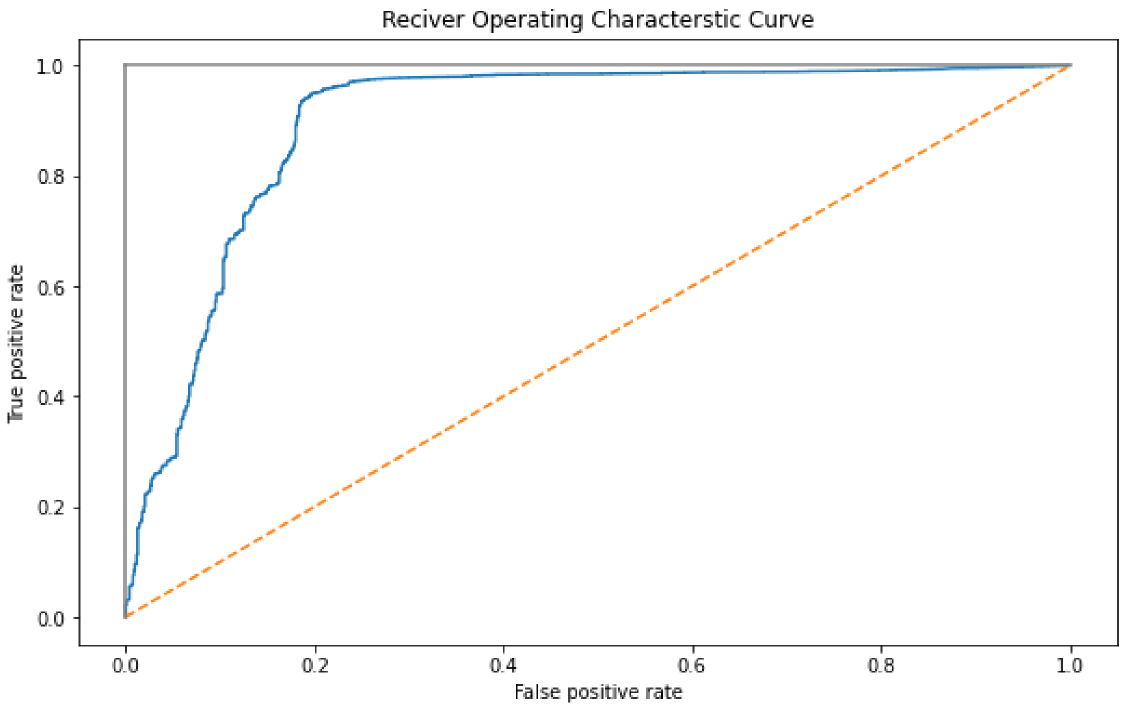

Figure 7.11 Logistic regression ROC AUC using applications features

Figure 7.11 and 7.12, show the AUC ROC curve for application feature set and global feature set respectively. We have found ROC for application feature set is 90.02% and for global feature set is 92.18% respectively. From the figure and ROC values we have found that logistic regression with global feature set are more promising for malicious application detection whereas applications feature set is not so efficient for prediction.

We have summarized the results for logistic regression in table 7.2 and compare the results of logistic regression with applications feature set and global feature set.

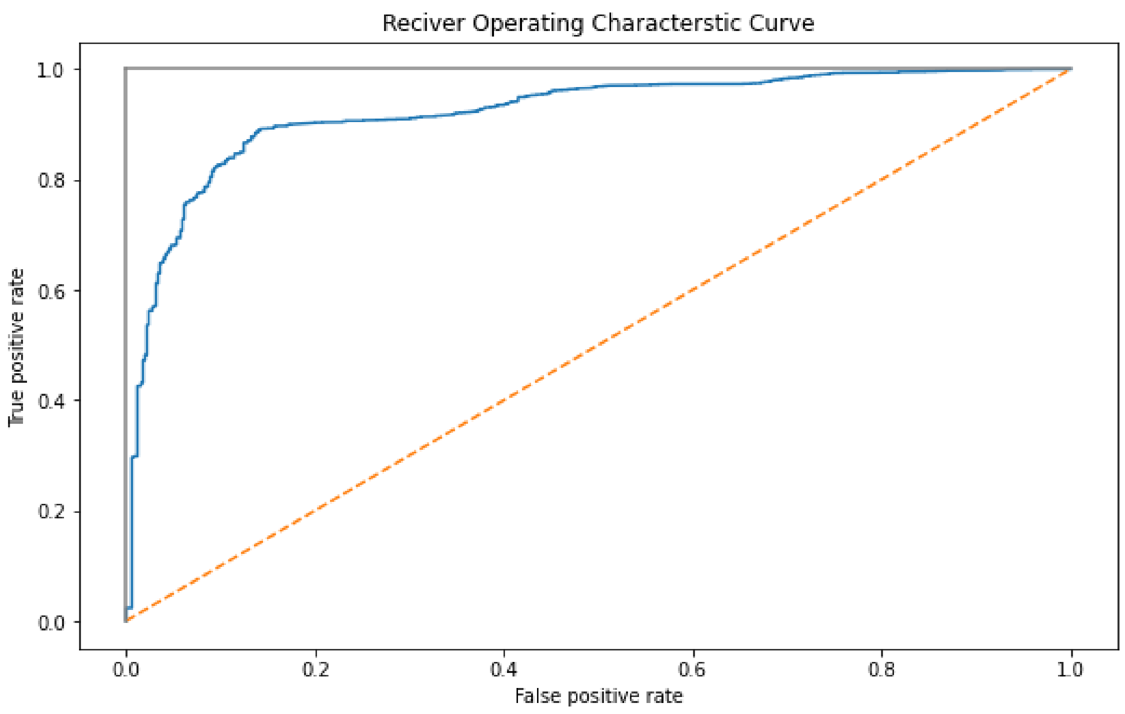

Figure 7.12 Logistic regression ROC AUC for global feature set for

Results for logistic regression		
Feature set	Applications features	Global features
Accuracy	84%	89%
F1 score	90.62%	94.01%
Recall	84.26%	90.38%
Precision	98.01%	97.92%
AUC	90.02%	92.18%

Table 7.2 Results for logistic regression

From table 7.2, logistic regression for application feature set has less accuracy the global feature set, however we are using F1 score to select the model. As we can deduced from table F1 score for global feature set is higher than application feature set. Precision is less quiet lesser for global feature set, but overall result parameters suggest that logistic regression perform well for global application features.

7.3 XGBOOST

The performance of XGBoost with both the feature set is described in this section. Figure 7.13 and figure 7.14 show the confusion matrix and normalized confusion matrix of applications feature set for XGBoost. We can clearly conclude from confusion matrix that True positive is 92% and False negative is 98% which means 92% of malicious applications predicted as malicious and 98% of benign applications predicted as benign where 8% is True negative and 2% is False positive which tells 8% malicious applications and 2% benign applications are misclassified.

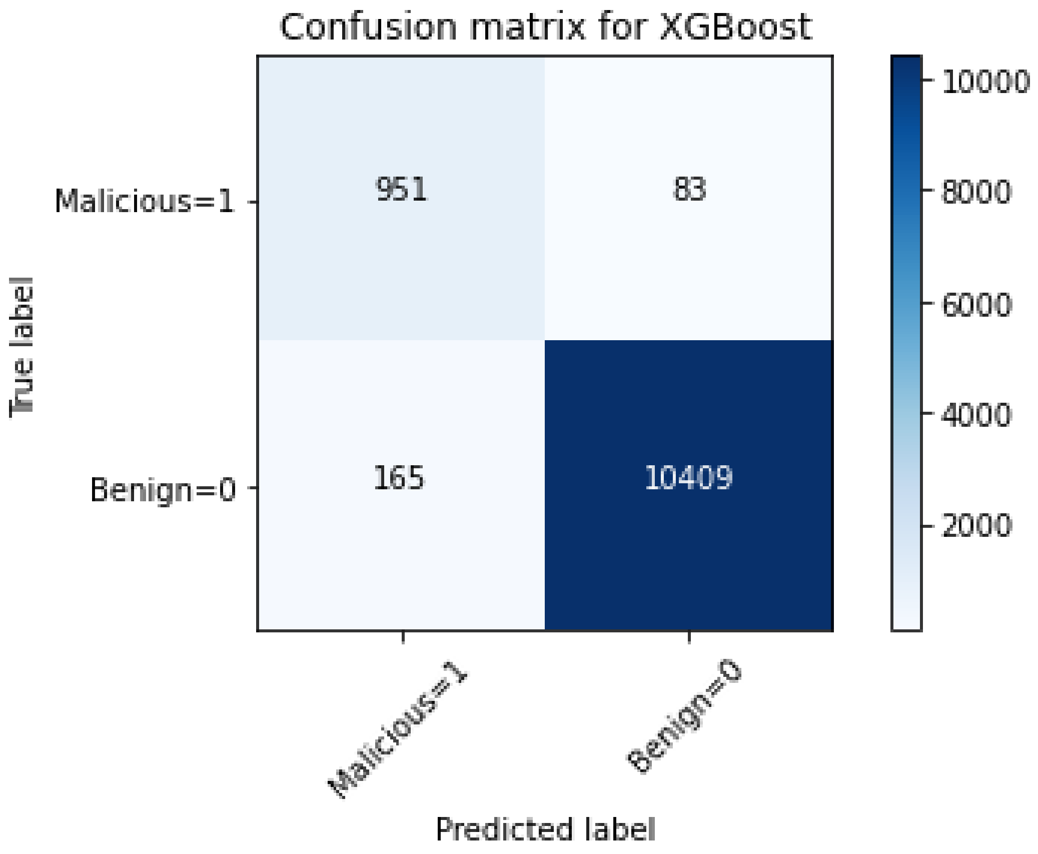

Figure 7.13 XGBoost confusion matrix using application features

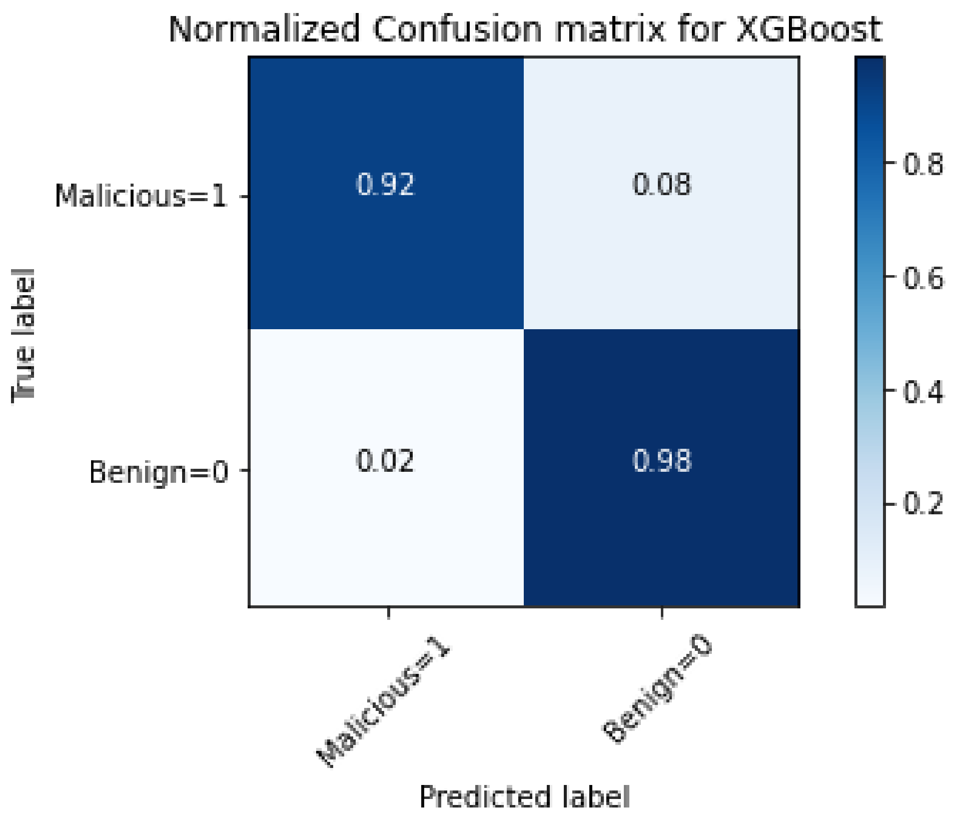

Figure 7.14 XGBoost normalized confusion matrix using applications features

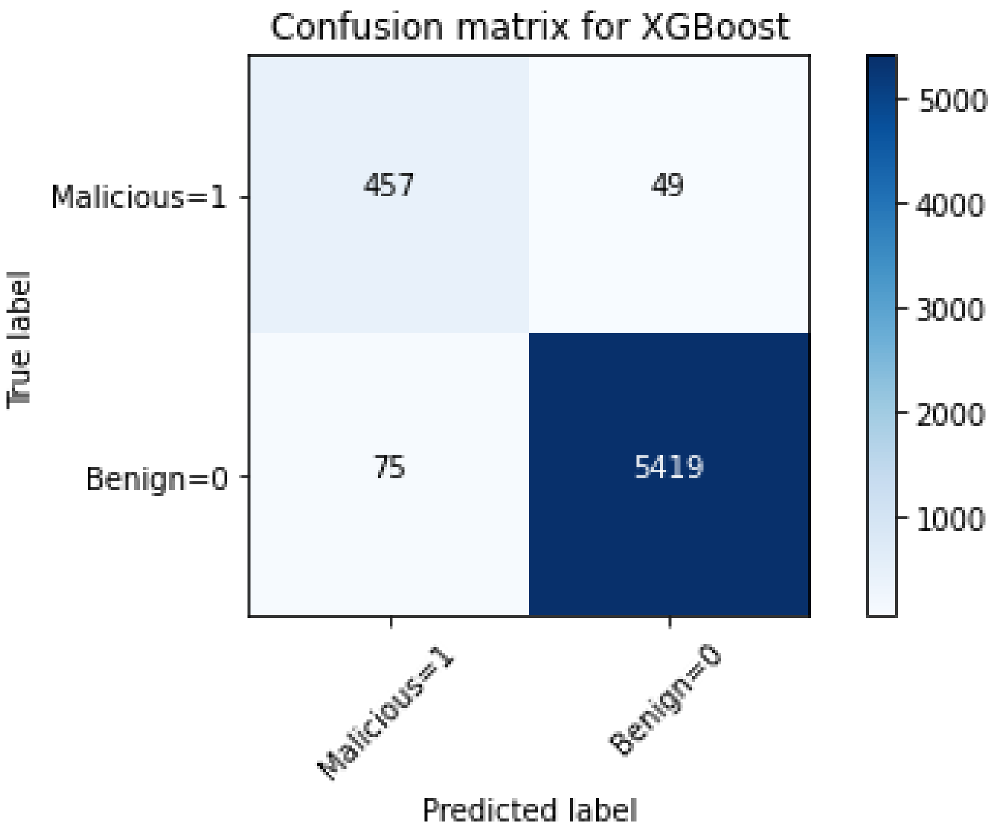

Figure 7.15 XGBoost confusion matrix using global features

Figure 7.15 and figure 7.16 for global feature set, we can clearly conclude from confusion matrix that True positive is 90% and False negative is 99% which means 90% of malicious applications predicted as malicious and 99% of benign applications predicted as benign where 10% is True negative and 1% is False positive which tells 10% malicious applications and 1% benign applications are misclassified. Here results are quite high, XGBoost is an extension of random forest which is a tree-based algorithm.

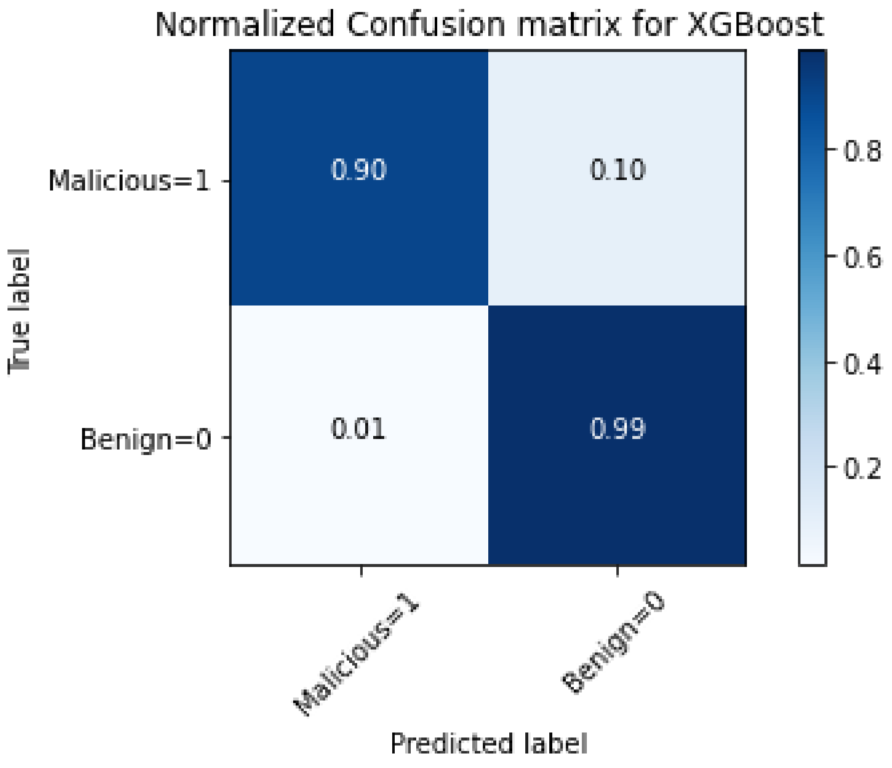

Figure 7.16 XGBoost normalized confusion matrix using global features

From the confusion matrix of both the feature sets there is not a noticeable difference, both exhibits almost same results. On analysing AUC curves for both the feature sets in figure 7.17 and 7.18, we can see that ROC curve for applications feature set is little better than the global feature set, we got AUC value for application feature set is 99.54 and for global feature set is 99.46. AUC tells us that XGBoost work better for application feature set than the global feature set. In table 7.3, we summarize the result for the XGBoost.

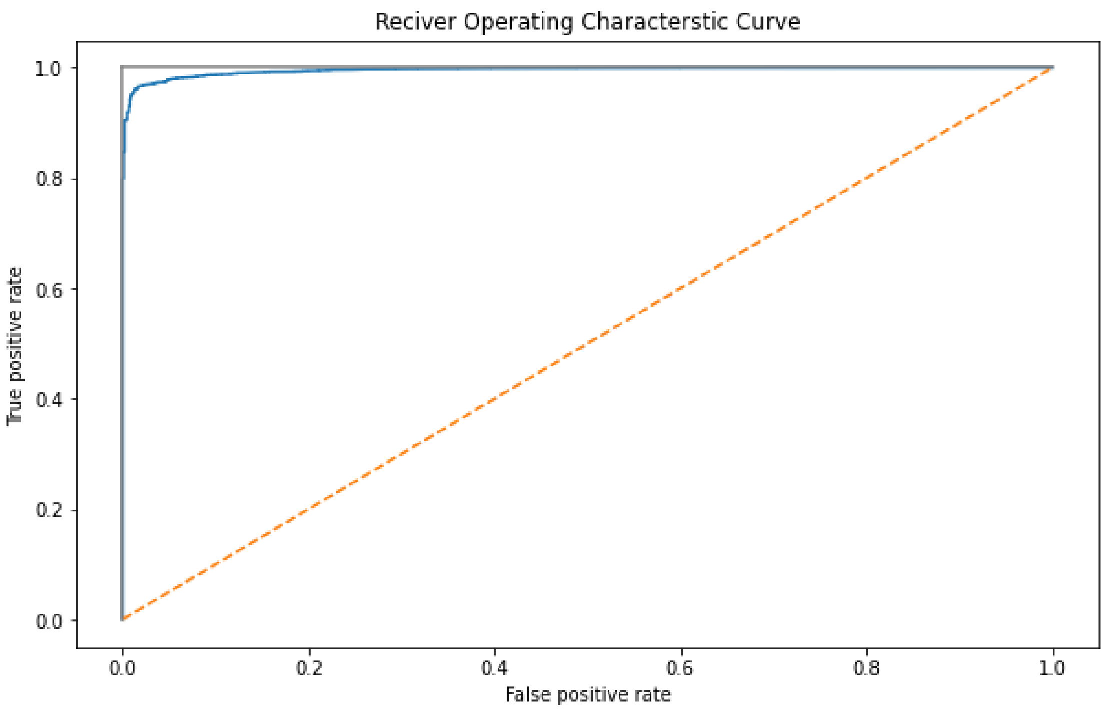

Figure 7.17 XGBoost ROC AUC for application features

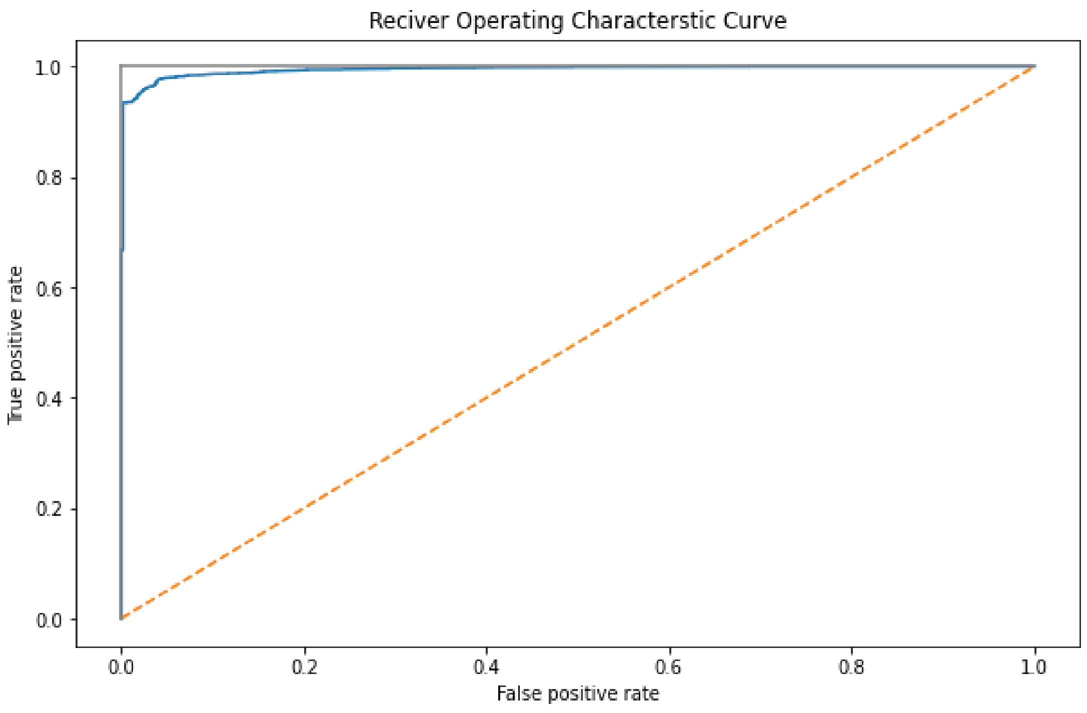

Figure 7.18 XGBoost ROC AUC for global features

XGBoost results		
Feature set	Applications features	Global features
Accuracy	98%	98%
F1 score	98.82%	98.86%
Recall	98.43%	98.63%
Precision	99.20%	99.10%
AUC	99.54%	99.46%

Table 7.3 Results for XGBoost

From table 7.3, we can see that result parameters for both the feature sets are almost same. Accuracy does not have any difference but AUC gives us the clear sight that the XGboost is better for applications features set then global feature set but F1 score says with a very minute difference that XGboost has worked good for global feature sets.

7.4 ENSEMBLE

The performance of ensemble with both the feature set is described in this section. Figure 7.19 and figure 7.20 show the confusion matrix and normalized confusion matrix of applications feature set for ensemble. We can clearly conclude from confusion matrix that True positive is 94% and False negative is 96% which means 94% of malicious applications predicted as malicious and 96% of benign applications predicted as benign where 6% is True negative and 4% is False positive which tells 6% malicious applications and 4% benign applications are misclassified.

Figure 7.21 and figure 7.22 for global feature set, we can clearly conclude from confusion matrix that True positive is 94% and False negative is 97% which means 94% of malicious applications predicted as malicious and 97% of benign applications predicted as benign where 6% is True negative and 3% is False positive which tells 6% malicious applications and 3% benign applications are misclassified. Ensemble uses voting mechanism to select the best result, here results are quite high and reliable.

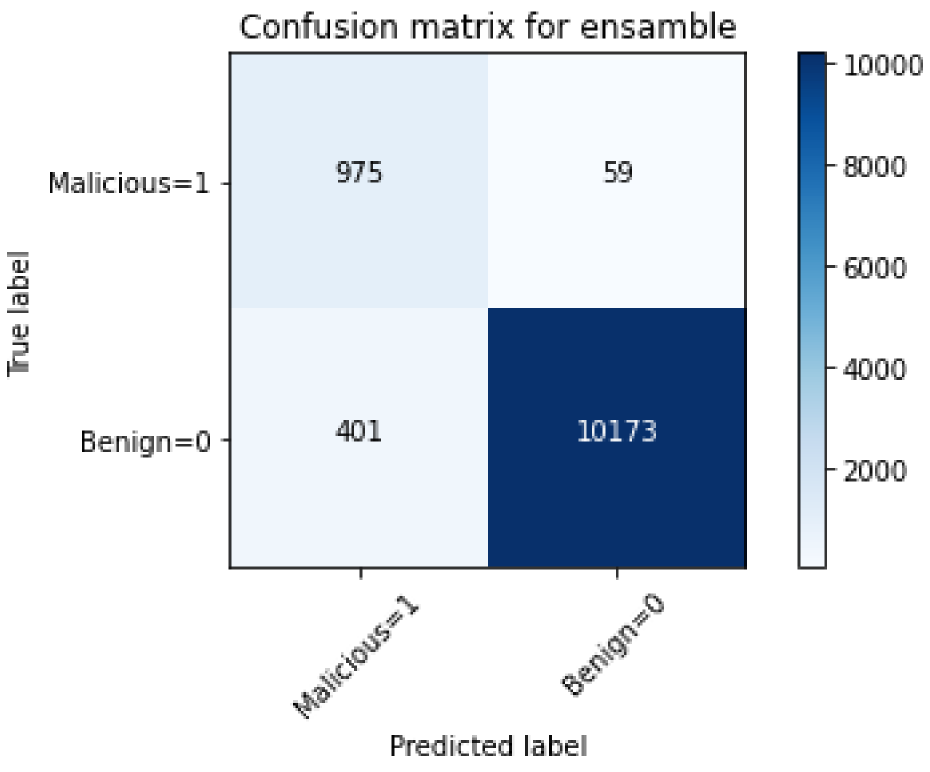

Figure 7.19 Ensemble confusion matrix using application features

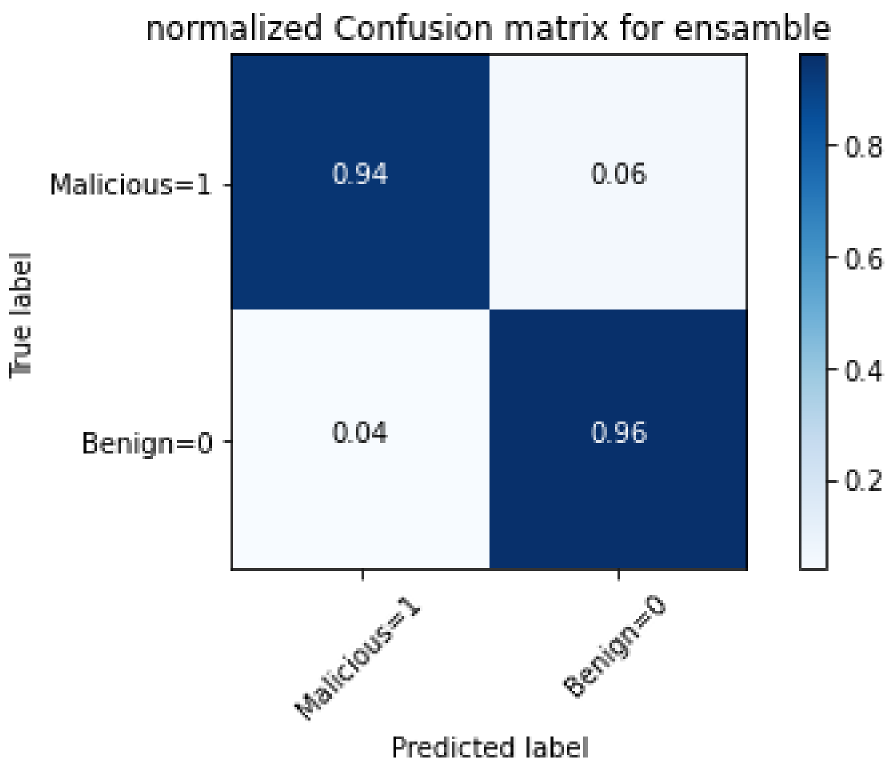

Figure 7.20 Ensemble normalized confusion matrix using application features

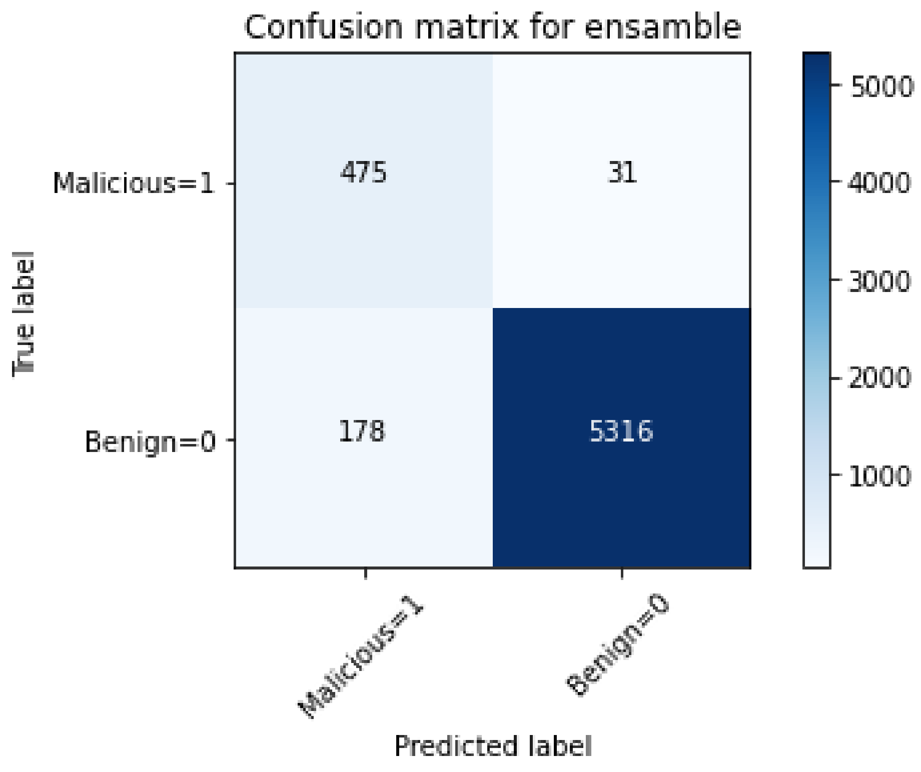

Figure 7.21 Ensemble confusion matrix using global features

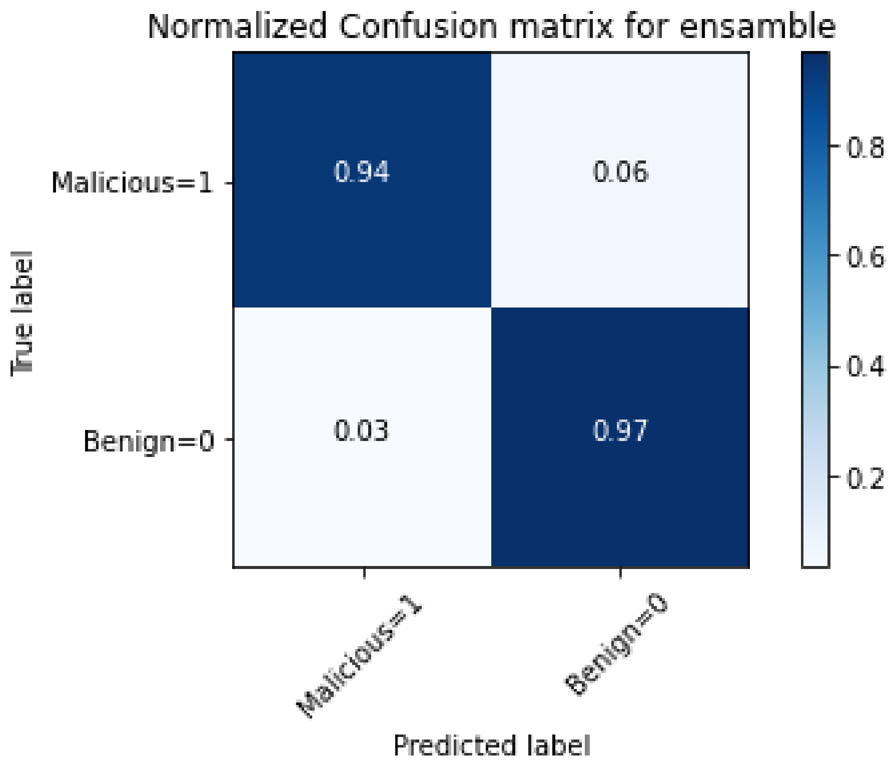

Figure 7.22 Ensemble normalized confusion matrix using global features

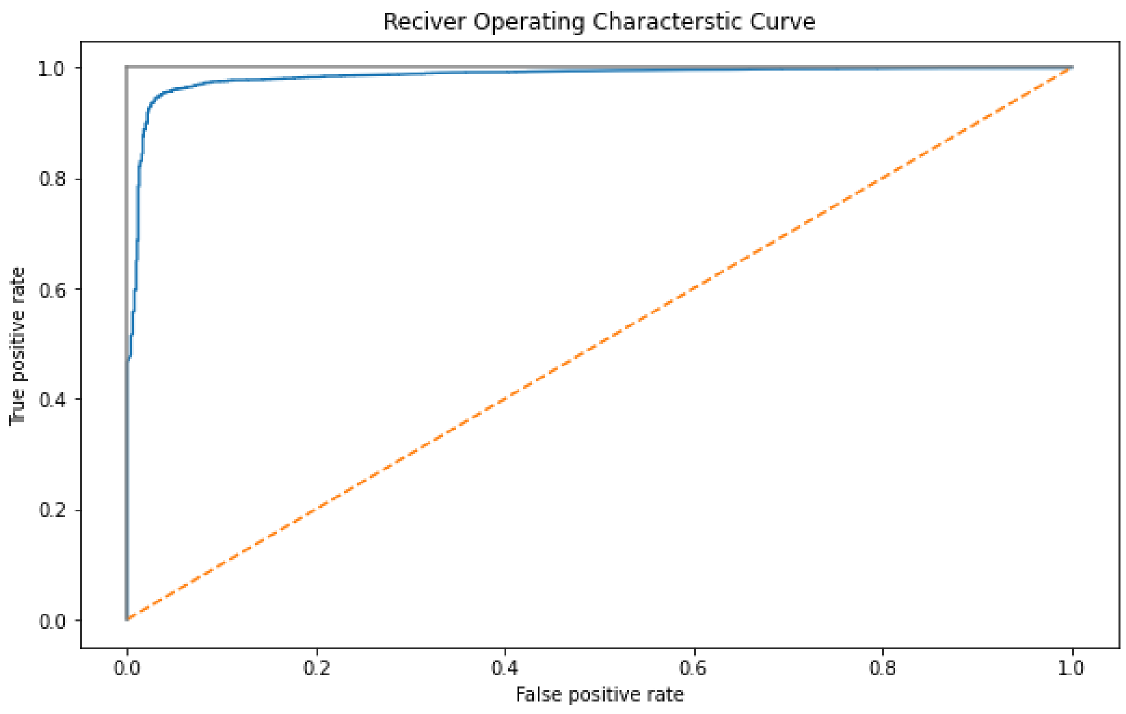

Figure 7.23 Ensemble ROC AUC for application features

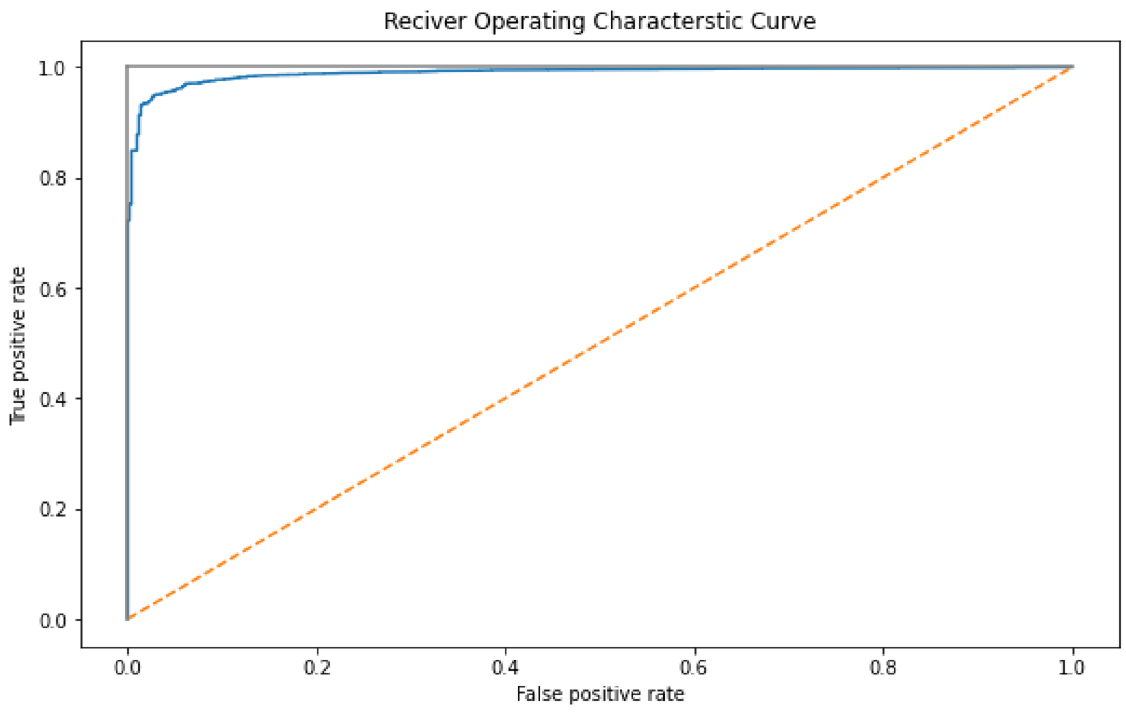

Figure 7.24 Ensemble ROC AUC for global features

Figure 7.23 and 7.24, show the AUC ROC curve for application feature set and global feature set respectively. We have found ROC for application feature set is 98.42% and for global feature set is 98.97% respectively. From the figure and ROC values we have found that ensemble with global feature set are more promising for malicious application detection whereas applications feature set is lagging quite behind.

We have summarized the results for ensemble in table 7.4 and compare the results of logistic regression with applications feature set and global feature set.

Ensemble results		
Feature set	Applications features	Global features
Accuracy	96%	97%
F1 score	97.78%	98.07%
Recall	96.20%	96.76%
Precision	99.42%	99.42%
AUC	99.54%	98.97%

Table 7.4 Results for ensemble

From table 7.4, we have seen that resulting parameters are almost same, there is not so much difference but ensemble work better with global features rather than with application features. Ensemble gives 97% accuracy and 98.07% F1 score but AUC is little lesser in the case if global features. As we are concluding our result on F1 score as F1-score is the harmonic mean of precision and recall so Ensemble work better with global features.

7.5 THE VERDICT

In this chapter, in preceding section we have seen KNN, Logistic regression, XGBoost and ensemble. These techniques we have selected on the basis of related work and our research for binary classification. We have seen the results for all four classifiers. In this section we have summarized all the results for all the classifiers and extract a verdict for better performance on the basis of F1 score. We have listed all the parameters in Table 7.5 for comparison.

Parameters	KNN		Logistic regression		XGBoost		Ensemble	
Feature set	Application	Global	Application	Global	Application	Global	Application	Global
Accuracy	97%	94%	84%	89%	98%	98%	96%	97%
F1 Score	98.58%	96.65%	90.62%	94%	98.82%	98.86%	97.78%	98.07%
Recall	9818%	95.90%	84.26%	90.38%	98.43%	98.63%	96.20%	96.76%
Precision	98.97%	97.41%	98.01%	97.92%	99.20%	99.10%	99.42%	99.42%
AUC	95.89%	87.27%	90.02%	92.18%	99.54%	99.46%	98.42%	98.97%

Table 7.5 Result comparison for classifiers

From table 7.5, we can conclude that XGBoost has performed best among all four classifiers. It has high accuracy for both global and application feature set of 98%. Moreover, as we are using F1 score for best performance metric so XGBoost has highest F1 score of 98.82% and 98.86% for application and global features respectively. Lastly, for AUC it is nearest to 1 that means XGBoost is giving highest probability of predicting correct results. So, The Final verdict we can state that for a binary classification problem like ours of classification and prediction of malicious and benign applications XGBoost which is a tree-based classifier has performed best and produces best result.

7.6 USABILITY

Usability obviously refers to how the model can be used, how the model can be deployed on a real device for malicious application detection. Usability of model are effected by three factors: i) train and test time, ii) model size and iii) memory usage. Lower values for these metrics show the low storage, less train, test time and low memory usage. We have two methods for deployment for this model either we deploy model on the device which require lower values for above metrics or we can deploy on the cloud but it will increase the latency as it will take time to import data to server, processing cost and then response goes to device.

7.7 LIMITATIONS

There are some limitations for this model which are as follows:

1 Dataset is collected from the Samsung galaxy S5, so it is not known that how the model performs on other devices.

2 For data collection self-written malicious applications are used. These applications are developed for research purpose only, so it is not known how model performs with real malwares.

3 We have found a smaller number of data points for some moriarty versions, so it could affect the results.

4 The method used for this research is behaviour based, behaviour analysis of the benign and malicious applications have carried out. On the basis of behaviour, we have drawn the results. In real scenario a benign application may use sensors like malicious application which leads to incorrect result.

CHAPTER 8

CONCLUSION AND DISCUSSIONS

In this chapter, we have conclude the findings of our research and discussed about the improvements which can be entertain in future.

8.1 FINDINGS

We have analysed the CPU usage, network usage and memory usage in presence of a malicious application and with benign application as described in chapter 5. We have seen the association plots for all three features and got an idea how the sensors are behaving in presence of malicious applications. As the malicious applications are self-written and behave like a real malware we can rely on the dataset because it is collected in real environment. We haven't used any emulator to mock the behaviour of real scenarios. We have seen that

1 CPU usage can be high or low with malicious applications
2 Network usage is high with malicious applications and low with benign applications
3 Memory usage is low with malicious applications and normal with benign applications.

We have analysed the performances of KNN, Logistic regression, XGBoost and ensemble method and find that XGBoost work best with global features resulting F1 score of 98.86% where other classifier are also very near to this score. We also find that AUC of XGboost is high with value of 99.58% with application features which tells that probability of predicting correct results is very high. There are 48 features in global feature set and 27 in applications feature set related to CPU, network, memory and battery. Our results have shown that our model can predict the malicious application with good accuracy. Moreover, XGBoost is a tree-based classifier which perform really good with our binary classification problem.

8.2 FUTURE WORK

There is so much scope has left to improve the detection method and to overcome the limitations as new methods and techniques can be introduce to increase the efficiency. For future we can entertain following points:

1. Conducting the research with real malware applications to improve the efficiency and reliability.
2. Data collection from multiple devices of different models so that a huge dataset from different plate-forms can be collected to analyse the real-world scenarios.
3. Dynamic detection methods can be improved in future research with real malwares.
4. Focus on optimize the resource requirements for detection methods, for training and testing time and memory usage.
5. We will deploy the model on real devise to test the model under real scenarios.

REFERENCES

[1] "total smartphone users in the world" https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

[2] "total number of apps on google play store" https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-playstore/#:~:text=The%20number%20of%20available%20apps,under%20the%20name%20Android%20Market.

[3] "Drop dialer" https://www.androidauthority.com/dropdialer-premum-rate-sms-malware-android-100783/

[4] "Google bouncer" https://www.theverge.com/2012/2/2/2766674/google-unveils-bouncer-service-to-automatically-detect-android-market

[5] "bankbot bypass google play stores bouncer" https://www.blackhat.com/docs/eu-17/webcast/10052017-scaling-security-operations.pdf

[6] "threats to android/ types of malwares" https://kaspersky.co.in/resource-center/threats/mobile

[7] "machine learning algorithms" https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

[8] "performance metrics" https://www.svds.com/the-basics-of-classifier-evaluation-part-1/

[9] "Association plot"
http://guianaplants.stir.ac.uk/seminar/resources/R_in_a_Nutshell_Second_Edition.pdf

[10] K. Patel and B. Buddadev, "Detection and mitigation of android malware through hybrid approach," in *Security in Computing and Communications*, vol. 536 of *Communications in Computer and Information Science*, pp. 455–463, Springer, Basel, Switzerland, 2015.

[11] Y. Mirsky, A. Shabtai, L. Rokach, B. Shapira, and Y. Elovici, "Sherlock vs moriarty: A smartphone dataset for cybersecurity research", in Proceedings of the 2016 ACM workshop on Artificial intelligence and security, ACM, 2016, pp. 1–12.

[12] Malware detection techniques "A Survey on Rise of Mobile Malware and Detection Methods"

[13] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y.Weiss, ""“andromaly”: A behavioral malware detection framework for android devices", Journal of Intelligent Information Systems, vol. 38, no. 1, pp. 161–190, 2012.

[14] M. S. Alam and S. T. Vuong, "Random Forest Classification for Detecting Android Malware", in 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, Aug. 2013, pp. 663–669. DOI: 10.1109/GreenCom-iThings-CPSCom.2013.122.

[15] H.-S. Ham and M.-J. Choi, "Analysis of android malware detection performance using machine learning classifiers", in ICT Convergence (ICTC), 2013 International Conference on, IEEE, 2013, pp. 490–495.

[16] A. E. Attar, R. Khatoun, and M. Lemercier, "A Gaussian mixture model for dynamic detection of abnormal behavior in smartphone applications", in 2014 Global Information Infrastructure and Networking Symposium (GIIS), Sep. 2014, pp. 1–6. DOI: 10.1109/GIIS.2014.6934278.

[17] J. Milosevic, A. Ferrante, and M. Malek, "Malaware: Effective and efficient run-time mobile malware detector", in Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress

(DASC/PiCom/DataCom/CyberSciTech), 2016 IEEE 14th Intl C, IEEE, 2016, pp. 270–277.

[18] A. Ferrante, E. Medvet, F. Mercaldo, J. Milosevic, and C. A. Visaggio, “Spotting the malicious moment: Characterizing malware behavior using dynamic features”, in Availability, Reliability and Security (ARES), 2016 11th International Conference on, IEEE, 2016, pp. 372–381.

[19] G. Canfora, E. Medvet, F. Mercaldo, and C. A. Visaggio, “Acquiring and analyzing app metrics for effective mobile malware detection”, in Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics, ACM, 2016, pp. 50–57.

[20] L. Massarelli, L. Aniello, C. Ciccotelli, L. Querzoni, D. Ucci, and R. Baldoni, “Android malware family classification based on resource consumption over time”, arXiv preprint arXiv:1709.00875, 2017