National College of Ireland

# Detection of Android Malware using Sensitive APIS's

MSc Research Project
MSc in Cybersecurity

Vivek Sajikumar
Student ID: 19220740

School of Computing
National College of Ireland

Supervisor: Ross Spelman

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | |
|---|---|
| **Student Name:** | Vivek Sajikumar |
| **Student ID:** | 19220740 |
| **Programme:** | Cybersecurity |
| **Year:** | 2021 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Ross Spelman |
| **Submission Due Date:** | 11/11/2021 |
| **Project Title:** | Detection of Android Malware using Sensitive API's |
| **Word Count:** | 5845 |
| **Page Count:** | 18 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** ……

**Date:** ……

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Detection of Malware using Sensitive API's

Vivek Sajikumar
19220740

**Abstract**

In this modern world filled with technologies, the usage and growth of android apps has gone up in the previous years. In a recent survey conducted it was noted that out of 100 randomly selected people, 87 of them used mobile devices based on android. This shows that android users are of 87%. As the number of users increases the chances of being targeted for malicious activities are also high. A survey states that an approximate of 38,000 malwares are being created by malicious users to exploit android devices. It is high time that, a measure to be implemented to overcome the malwares. As technologies are being improved on the daily basis and by using reverse engineering methods, malicious users are generating malwares which are extremely difficult to detect. Malwares in a device can be broadly classified as code based malware and behaviour based malware.

This paper proposes a malware detection method which is based on behaviour. This method is considered to be effective and agile. The proposed method is based on sensitive API. There are two phases in this method, which are training phase and testing phase. The training phase is responsible for collecting the API of multiple apps in a eigenvector format. This collected data are then used later on to find out the connection between API calls and malware, from which new combination of API are generated. Performance of each algorithm are tested and the best 3 are selected using the help of K-fold cross validation method. The selected models are then embedded to ensemble learning model for the evaluation of performance. Performance of the model is identified by its working efficiency and its outputs accuracy. In this paper, the algorithms used are Neighbour's, SVM Classifier, Decision Tree, Naïve Bayes, Random Forest, Linear Discriminant Analysis and the top 3 performing algorithm along with ensemble learning model gave out an accuracy rate of 96%.

# 1 Introduction

Mobile devices in this fast-moving world have become one of the essentials of life. It has become a part of a routine for all of us. Mobile devices running on android platform are being widely used than any other mobile device platforms. This is due to the easiness by which you can use the phone as well as the option given by the device to make changes to its properties which allows a user to customise their phone according to their wish. [1] The number of android users has tremendously increased, which in turn increase the number of malicious users trying to exploit the devices. Many malicious applications are being developed and uploaded to the play store and are available on the internet. User unknowingly download the applications and the malware gets activated in the device.

The system resources are being managed by the API resources. The number of API's differ in each application, in which high risk APIs are mostly seen in malicious application. By observing a malicious application and a genuine application, sensitive APIs are being used or called often in a malicious application than a genuine application. This observation can be used to determine whether an application is malicious or legitimate. This will not be an easy task ass each application will have a large number of API used to run. So, finding a set of APIs to keep as a standard reference for future investigations or findings for detecting malware will be difficult task.

This paper focuses on identifying malware in an android device using behaviour-based method. This method consists of two phases namely, detection and training phase. This can be further categorised. A combinational set of 20-dimensional API are being categorised based on the sensitivity level from the mutual information. The output is predicted by combining the output of the KNN with decision tree and the results were observed to be 92% accurate.

This paper focuses on detecting malware in an android device by using its sensitive APIs from the application. The mutual information model was taken into usage for the development of a refined set of sensitive API calls. Moreover, each algorithm was closely examined for their accuracy rate and found out that the top 3 algorithm with highest accuracy rate was found to be K-Neighbour's, Decision Tree and Random Forest with an accuracy rate of 93%, 95% and 95% respectively. These were then embedded to ensemble learning model for the purpose of detecting malware and was found out that the accuracy rate was around 96%. Which proved that detection of malware using this method was more accurate and efficient.

# 2 Related Work

Observing a literature survey, it was seen that almost three fourth of the analysis made was based on static analysis. There must be more work to be performed on these methods to get a better and efficient method for detecting malware in a device. This must be accomplished by obtaining a method with higher accuracy with the minimum number of features used as the number of features increases the time required for testing and training also increases. [2]

The [3] proposed that a method in which the permissions were taken from the AndroidManifest.xml file. For the filtration process three levels of data pruning was performed which mainly focused on clearing out the extra permissions which were not required without affecting the accuracy. The results were as good as the accuracy rate given by google which

gave out the dangerous 24 permissions. The accuracy calculated was around 91.34% and a drawback found was that it does not work on skewed dataset optimised to 22 permissions.

[4] proposed a method in which, analysis was made based on the permission in the Android manifest of the application and observed the most relevant 37 permissions using Chi square method with the help of Naïve Bayes classifier. The accuracy was observed to be 82.4%.

[5] proposed a method in which for detecting malware in an application based on the traffic in the network. Chi square test along with Information gain were used to rank the selected features. Naïve bayes classifier was used for the classification in this method and the optimization was done to the method which reduced the feature to 9.

[6] suggested a method called Mmda, which was developed on the basis of the metadata present in the android manifest file. 122 features were being extracted from the application and different classifiers were used to check the accuracy rate and finally ended up in Random Forest classifier which gave out 94% accuracy. Permissions, hardware feature, total declared actions were some of the extracted resources.

[7] proposed a method in which, the detection of malware in an android device was carried out through the category of the application. They categorised legitimate application and malicious application with respect to their features. For this method they have used SVM along with 2775 features.

[8] proposed a method in which DEX string along with android manifest strings and Hashes are being used. Using dimensionality reduction method, they reduced around 10 million features to 10,000 and classified using SVM to get an outstanding accuracy of 99.91%.

[9] came up with research called ScanMe by which the application will be deployed to the cloud platform and malware will be detected using a detection method based on cloud and a report will be generated about the uploaded APK file.

[10] proposed a cloud-based framework which could evaluate the risks that can be caused at the API level called ASCAA. The evaluation was done for the permissions in the android manifest file. After a bit of research and study, it was concluded that ASCAA was able to determine whether an application was malicious or not as the failure rate was one is to eight. This method had an accuracy rate of 93% but was not fully reliable as one out of eight went wrong.

[11] made analysis based on static code. The behaviour of the malware in an application were captured using its authorizations in the application as well as the API calls. This was carried out using KNN and Naïve Bayes classification model. This method gave out an accuracy rate around 95% with a true positive rate of around 89%.

[12] proposed a method by which detection of malware in an android device took place using static analysis along with machine learning. This method was considered to be fast as the proposed method was able to analyse 10,000 applications in less than 24 hours, which means each application took an estimated time of 750 milliseconds to run. For static analysis, the eigen vectors were put into join vector spaces. Although with this many advantages, this method lacked dynamic analysis.

[13] proposed a method by which pairing of malicious profile along with similar data flow takes place. Mainly, two steps are being followed in this method. Firstly, malware profile and the information flow are being checked for matching. Secondly, the container node is being checked to find out whether the source node and the destination node corresponds. This method was only applicable on Nexus devices. On the other hand, our proposed method works perfectly on all the android versions as well as the devices.

For finding the non-legitimate code in android application, [14] introduced a method using machine learning. The mostly used APIs in a legitimate and malicious applications were noted and ranked. More than 60,000 applications were then assessed using Random Forest classifier from which the top 50 genuine and malicious API were chosen using the ranking. It was seen that, detection rate for finding a genuine application was around 99.98%.

[15] introduces a method which could overcome the limitations caused by static and dynamic analysis. The time-consuming process of static analysis for identifying the use of API along with the intent messages with time complexity of the dynamic analysis can be resolved using this method. According to this method, the Dalvik bytecode was being converted in RGB files. Using the help of CNN method, the model was being trained for detecting malware in android application. It was observed that the detection of malware from an application took around 0.22 seconds, which was very fast compared to the previous methods. This method was fast and gave out an accuracy rate of 93%.

[16] proposed a method in which the malware which are there in the application are being converted into greyscale images. For extracting GIST descriptor have been used. The .DEX will be extracted from the android application and will get converted to the 8-bit binary format. These are then inputted to machine learning for training and testing the dataset. These are inputted to KNN, DT and RF classifiers and checked for accuracy. It was seen that, none of the classifier gave out an extremely good accuracy rate.

| Author | Referenced Characteristics | Drawbacks |
|---|---|---|
| [3] | Multilevel data pruning to optimise permission<br><br>Accuracy was 91.34% | Does not work on skewed dataset |
| [4] | The permission gathered were ranked using chi square.<br>It was optimised to 37 permissions<br>Its accuracy was 82.4% | |
| [6] | Optimised to 122 features<br>Accuracy was 94% | |
| [8] | Accuracy rate was 99.91% | The feature vector was too large |
| [9] | Statistic and dynamic analysis performed by cloud based sandbox<br>Accuracy 86% | |
| [10] | Accuracy rate was 93% | One eight of the results failed |
| [11] | For finding the true positive rate the accuracy rate was 95.1% | True positive rate accuracy was 89% |
| [12] | Lower false alarm rate<br>Very quick in detecting malware | No dynamic analysis |
| [13] | | Detection of malware limited to only Nexus smartphone |

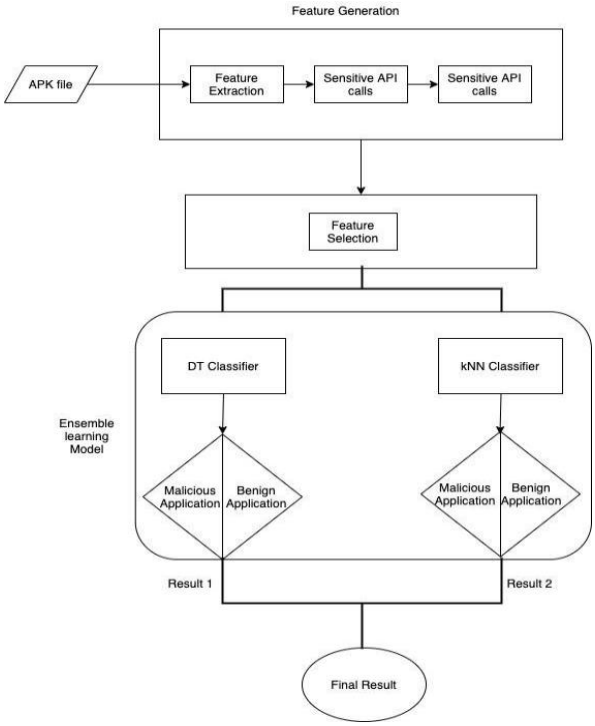
# 3 Research Methodology

The main reason for implementing this proposed work is for achieving a method for detecting malware in android devices with high accuracy rate as well as with the help of minimum number features. This paper mainly comprises of two parts. These can be categorised into training phase and detection phase. Also, the learning model comprises of feature generation as well as ensemble learning model. The dataset has been chosen from Kaggle which comprises of 215 vector attributes which have been collected from more than 15,000 applications.

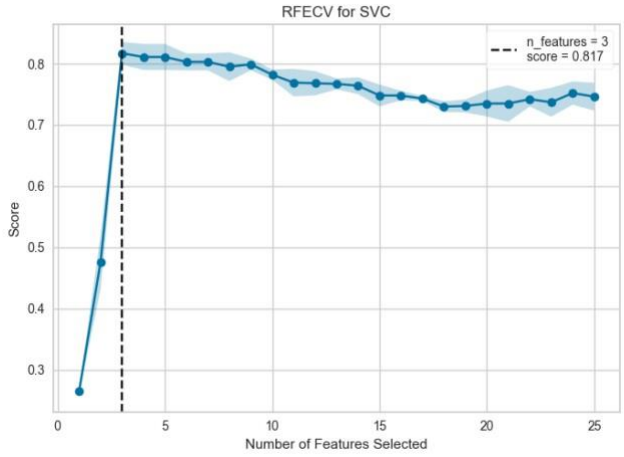
## 3.1 Recursive Feature Extraction

Recursive feature extraction is the process or the method by which the features are being selected so that each feature fits into a model. This is done by eliminating the features which are weak until reaching the specified number of features for the model. This type of feature

extractions is called as recursive feature elimination. This method consists of two attributes namely coef_ and feature importance_. Among all the features available in a model, it is uncertain to predict which all features are required, and which features are unnecessary. Recursive Feature elimination provides the model with the required features. The best performing arrays of features are chosen with the help of K-Fold validation and RFE. The cross validated test score and variability are then displayed by the RFECV visualiser.




Let us take an example into consideration to understand the working of recursive feature elimination. Take a dataset with 25 features in it. Out of the 25 features in the dataset, consider only 3 features as genuine. From the figure, the curve forms a curve by increasing while reaching the value 3 and then the curve descends after the value 3. This shows that those value are not required for the model. The cross validated portion is being represented using the shaded portion in the figure.



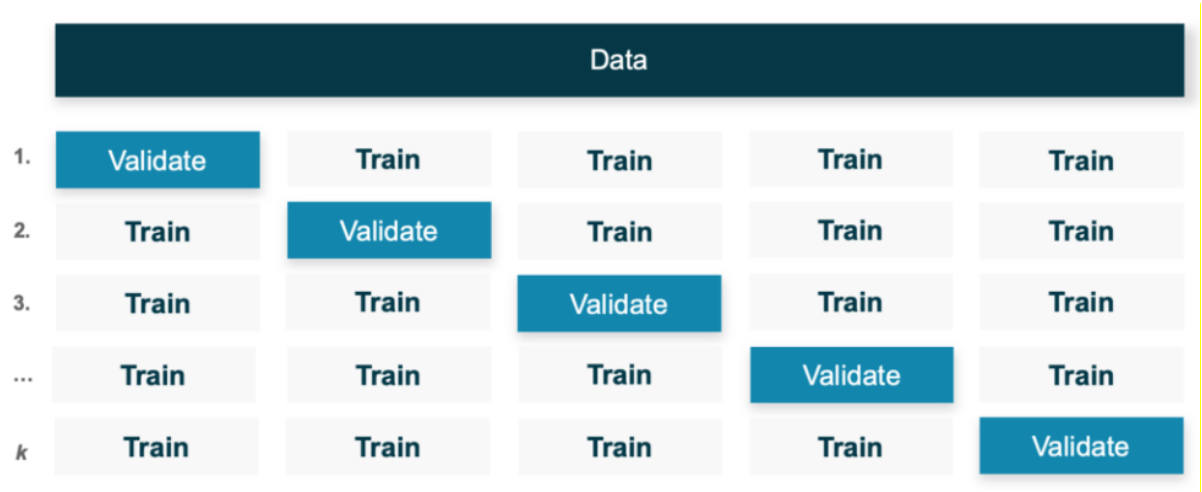

## 3.2 Optimization

Feature optimization can be achieved to improve and increase the model's efficiency and performance by keeping the features which are only required by bringing in an extraction method. In the base paper, the author has not mentioned anything about how accurate the learning models are for the process of detecting malware in an application. The paper does not give any information to support that the algorithms used gives out the best performance or accuracy. The author has randomly chosen algorithms to detect malwares and it was found that, this model gave out an accuracy rate of 92%. It is not necessary that, the randomly selected classifiers along with ensemble learning model will give out the best output accuracy rate for detecting malware. In-order to get the best result, all the 6 classifiers will be individually trained and tested to get the best output.

The widely used classifiers are, Naïve bayes, Random Forest classifier, KNN, SVM, Decision tree and linear discriminant analysis. All the mentioned classifiers are being tested separately along with K-Fold validation method, and the accuracy rate of each model are compared to find out the 3 best performing classifiers. These are then inputted to ensemble learning model.

## 3.3 K-Fold Cross Validation

K-Fold validation method can be categorised to testing phase, training phase and then validation phase. This method is used as it is considered to be less biased compared to other methods. The number of folds to be done on the dataset in a K-Fold validation method is represented by the letter k. There will be a certain number of fixed turns(k-1) for the training set. Using the observations, the model will learn about the distribution in each fold. The value of K is usually not too high or not too low, it is set somewhere between 5-10 as low value leads to highly biased model and high variance can be caused if the k value is too high. This method is termed as Leave one out CV.




https://www.r-bloggers.com/2019/10/evaluating-model-performance-by-building-cross-validation-from-scratch/

The average performance of the model is calculated after every iteration of the K-fold model made by the k value. These values are then returned back to the model. In each loop, a new combination of data are created as the value of the validation and training data are modified in each loop. The best model is noted down in each loop, this is performed using a call back function. The saved files are named differently in each loop in-order to prevent overwriting of data. For the purpose of evaluation, weight of the model is taken into consideration. For evaluating the performance, the function is being inputted with the highest weight of the model.

The models performance is calculated using K-Fold validation and the average is calculated. The best set of features will be taken for the implementation part. Ranking of the API takes place with respect to the malicious behaviour of the API. With respect to the reference standards twenty-dimensional vectors are being used. These vectors are then inputted for training to ensemble model.

## 3.4 Feature Vector

The process of creating sensitive API is a time-consuming process, to avoid this problem, for this research project a dataset was taken from Kaggle which comprises of application both malicious and legitimate with a set of APIS within them. This dataset was chosen because, the sensitive scores of the API are pre-recorded and placed in the dataset. From the dataset it can be clearly understood whether an API has been used in the application or not. For example, if the API has a value of 0 then that API has not been used in that application. Similarly if the API has a value 1 then it means that that API has been used in that application.

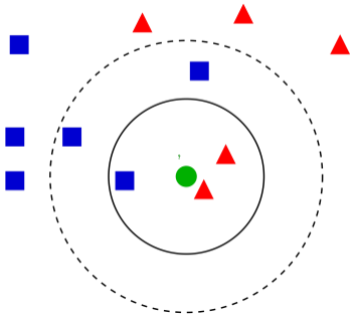
The API present in the application are represented or displayed in the format [X1:Y1;X2:Y2;X3:Y3;...Xi:Yi;...Xn:Yn](i=1,2,3…n) which is a comma separated value file, in which Xi shows us the name of the API and on the other hand Yi shows us the binary value.

From the given 20-dimensional, twenty sensitive API will be inputted to the classifiers. After observing the K fold cross validation carefully, the best will be embedded to ensemble learning model.

# 4 Design Specification

### KNN

KNN can be termed as a basic machine learning technique that can be used for classification. This classifier method distributes data to classes which are showing similar properties. The performance of this classifier is mainly dependent on the k value. To archive a greater accuracy, the K value should be set to a higher value. The square root of the recorded observation in the dataset are used in-order to find the most accurate k-value



### SVM

SVM is a form of machine learning method, that have been widely used for novelity detection, finding regressing and also for classification. This strategy is relatively new and is formed or based from the statistical learning theory. Because of its high accuracy rate, SVM has become famous for multi-classification problems. This method also outperforms the currenly existing techniques as it has a 2 dimensional description of optimal surfaces which are being derived from linearly separatable examples and also SVM is dependent on the principles of structural risk management.

### Decision Tree

Decision tree algorithm is another form of classification, which is developed by closely studying the dataset. The data features are being referred by sample questions which are from the data categorization as each question is being designated to a child node. The categorisation takes place from the highest rank to the lower rank. Interpretation can be seen as a better option in DT when compared to neural network.

### Naïve bayes

This algorithm works with simple probability classification technique, which works on assumption of all features are being not dependent. The disadvantage of Naïve bayes is its independent data although its efficiency is comparable with neural networks and decision tree.

However, extending its structure will help in avoiding this problem. Mostly, this method is used for text classification in ML.

**Random forest**

In order to increase the classification accuracy, ensemble classification is used, which is known as Random Forest. When comparing to other classification, the accuracy of the results provided by this classification us higher as it employs many models to tackle a problem. The result for a problem is found out by a voting system and the final choice is being made with a prediction of majority votes. Because of the increased diversity, it is seen that the accuracy improves over the time.

**Linear discriminant analysis**
For the purpose of reducing the dimensionality in ML pre-processing steps and for classification of patterns Linear discriminant analysis is widely used. In-order to increase the class variance within the class LDA improves the separability of class by translating features into a reduced dimensional space. This can be both dependent or independent from the class. If the size of the dimension increases and overvalues the sample in the matrix then this method will become inefficient.

# 5 Implementation

For detecting malware in an android device in this research project all the classifiers have been independently checked for their accuracy rate, from which the top three performing algorithms have been chosen and embedded to the ensemble learning model. All the required information about the implementation part that have been carried out in the research are mentioned below.

## 5.1 Environment Setup

**Python**
The coding for the research project model was executed in python version 3. Unlike other programming languages, because of its simplicity and readability this programming language was taken into consideration for doing the project. Moreover, even beginners can build an application using python language as it is simple and straightforward. This programming language also provides users with libraries that provides a wide range of library which helps users to save a lot of time.

**Anaconda Prompt**
The main advantage of using anaconda prompt in your application or project is that, once you have installed a python version in your system, this prompt lets you build a virtual environment that can be accessed from sites that have python in it. Anaconda prompt also gives the user a wide set of libraries which are prebuilt, and which helps them to develop easily. A lot of time gets saved by using the pre-built libraries.

**Tensor flow environment**
Tensor flow environment is mainly used in this project for importing the packages from python. Data flow graphs can be created using tensor flow with the help of the open-source AI library

which have been provided by the Tensor flow ecosystem. This also helps the user to create a huge number of neural networks along with other layers.

## 5.2 Dataset

The process of creating sensitive API is a time-consuming process, to avoid this problem, for this research project a dataset was taken from Kaggle which comprises of application both malicious and legitimate with a set of APIS within them.


https://www.kaggle.com/shashwatwork/android-malware-dataset-for-machine-learning

The selected dataset comprises of feature vectors from the application. The dataset has been created by combining 15,000 applications. This combination of application includes legitimate apps as well as malicious one too. Out of 15,000 applications in the dataset 5560 application are not genuine ones, and the rest are genuine. This dataset was mainly developed for detecting malware in devices running on android platform.

## 5.3 Dataset Pre-processing

As the processing part for this project, the data is closely observed and checked to find if any of the values are found to be null or empty. If any values are found to be null or empty, then that field is being replaced with the value 0. After all the empty fields have been filled with 0, then the dataset is split into data and labels. Like filling empty slots with 0 value for the data, if any of the label field is found to be empty or null it is being replaced or places with the value “B”. B denotes the API is normal and S tells the user that the API is malicious.

Once all the fields including the data and label have been filled and made sure that there are no null values, all the label values are converted to binary format. S value gets converted to 0 and B value gets converted to 1. After these steps, once all the data present in the dataset now have been converted to integer type, this dataset will be converted to a NumPy array

## 5.4 Feature Extraction using Recursive Feature Elimination (RFE)

The RFE model was defined and fitted to the pre-processed data and labels. This was considered as the first step for feature extraction using RFE. This method then filters out all the unwanted elements and only the important elements were chosen. In this project, 20 features were chosen from the dataset.

## 5.5 Splitting data and label

Splitting of the data and the label into the training set and testing set is done using the K-fold method. Before splitting the dataset into the above-mentioned set, the dataset is randomly shuffled. Once the shuffling has taken place, the dataset is being separated to k number of groups. From the formed groups by randomly shuffling, a set of data is taken for the purpose of testing and the rest is given for training. Following this, a model is being made to analyse the training set and are analysed on the testing set. Each time the score will be calculated and recorder and the model that have been analysed will be removed. Once all these steps are done, the score of the sample model is being used to summarize the skill of the model.

## 5.6 Training and Testing

During the training and testing phase, each of the classifiers KNN, SVM, DT, RF, NB and LDA have been trained using the data and the label that was there in the dataset. Once the training was completed by each classifier, each of the results would be saved for future references. After the completion of training the data, the same data would be used for testing. Finally, the performance and accuracy of the trained model was calculated by comparing with the model set by testing.

## 5.7 Ensemble model

Ensemble is a ML learning technique, which can be used to combine many base models for producing one optimal predictive model. After the completion of training and testing of the data in the dataset, now comparison of all the models is being made with their accuracy rate in output and the models with the most accurate outputs are being selected. In our project, the selected 3 classifiers were KNN, DT and RF. In-order to come up with a optimal predictive model, all the three classifiers were embedded to the ensemble learning model for training and testing. The predictions were made by the model, the dataset was being trained and tested and finally the performance was being calculated. It was seen that; the model gave out and accuracy rate of 96%.

# 6 Evaluation and Discussion

For this project, an execution environment was setup for carrying out the proposed work, which had a computer system with 8GB RAM and more than 500 GB storage running on windows 7 or later. This approach was executed by coding in python programming language.

## 6.1 Dataset

The sample dataset with malicious application and legitimate applications was collected from Kaggle.

## 6.2 Result Analysis

In order to measure the accuracy, TN, TP, FN and FP were calculated using the formula

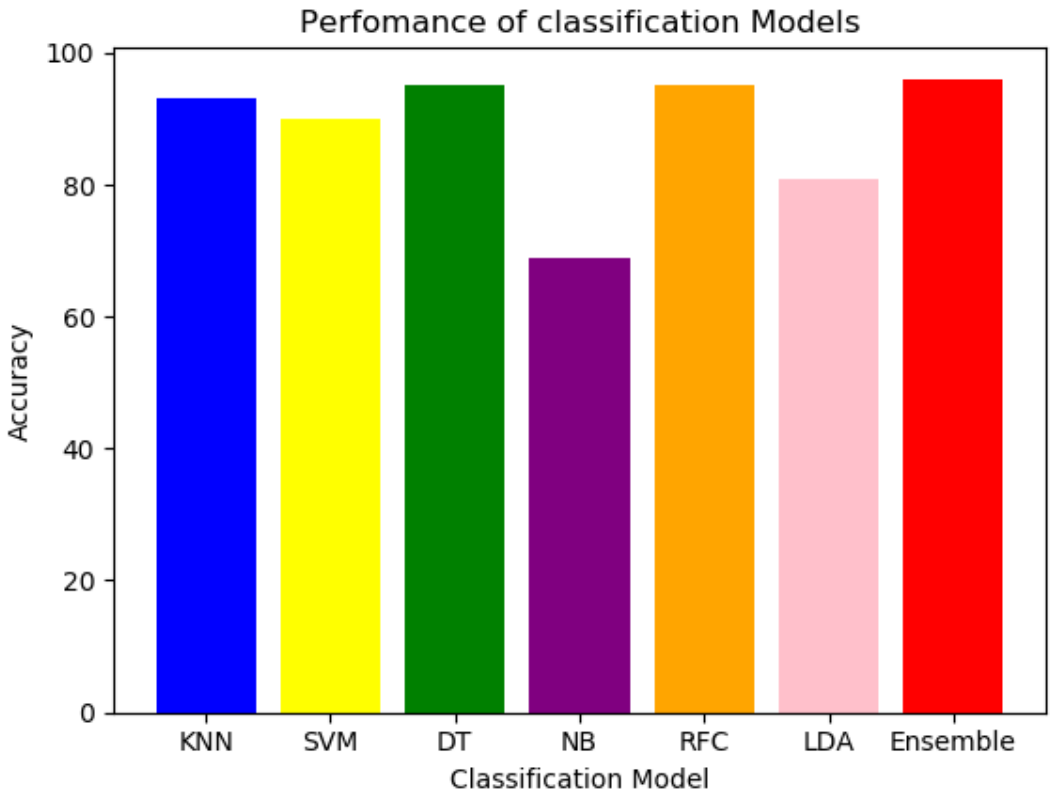
in which True positive is represented as TP indicated all the correct samples represented as malware. TN represents the true negative value which are the correct samples represented as legitimate, FP represents false positive which means all the samples that are incorrectly shown as malware and finally FN which represents false negative, the number of samples which are shown incorrectly as legitimate apps.

Accuracy= (TP+ Accuracy=(TP+TN)/(TP+TN+FN+FP)

All the models were separately used for training and testing the dataset and individually accuracy rate of each model was being recorded. In-order to bring the optimal solution from ensemble learning model it was necessary that the top three performing classifiers with the dataset must be found. Testing and training were implemented, and the accuracy rate of each model are being listed below.

| Model | Accuracy rate |
|---|---|
| KNeighbors | 93% |
| SVM | 90% |
| DT | 95% |
| NB | 69% |
| RF | 95% |
| LDA | 81% |
| Ensemble | 96% |

The accuracy rate for each of the model have been shown below in a graphical representation




The goal of the project was successful as the base paper used 2 randomly selected classifier and the output accuracy obtained from the model was 93%. Our proposed method clearly shows that Ensemble learning model along with KNN, DT and RF have boosted up the accuracy rate from 93% to 96%. This shows that the predictions that was made in the starting of the project was correct and the proposed model was able to give out an application which can be used to detect malware using sensitive API with more accuracy, performance and efficiency.

## 6.3 Discussion

From the above representation and calculation, it is clear that ensemble learning model when embedded with KNN, RF and DT brings out an accuracy rate of 96%. This proposed model now can be used in the future for detecting malware in android devices or can also be used for separating applications which are malicious or legitimate. This model has a better accuracy rate than all the other classifiers because, each of the models were separately trained and tested with the dataset and accuracy rate of each models were noted down. In-order to get the optimal accuracy from these models, ensemble learning model was inputted with top 3 performing classifiers. The model was then trained and tested with the dataset and the dataset was optimised to get a better performing accurate and reliable working model. In the base paper, feature elimination was not performed, in-order to boost up the efficiency and performance of the model, recursive feature elimination was included in this model with the help of K-Fold cross validation. This helped in removing the weakest features in the dataset.

# 7 Conclusion and Future Work

In this modern world, the usage of smartphones and smart devices are increasing on daily basis. As the number of smartphone users increase, the risk of being attacked by a malicious user is also increasing. As the technologies grow every day, it is becoming very difficult to figure out whether an application is malicious or genuine. In order to save all the users from the malicious attacks that have been performed by attackers it is high time that a method to be introduced so that no more users can be trapped into the hands of a malicious user. This method lets the user know whether the app is safe or not and by giving the user a chance to decide whether to install the application or not. The main aim of this project was to build a software which was reliable, accurate and easy to use for detecting malware in an android device. For the purpose of increasing the accuracy and to get the optimal solution from ensemble learning model, all the classifiers were individually trained and tested using the dataset and the top 3 classifiers namely KNN, DT and RF were select and embedded to ensemble learning model. It came out to be more efficient than any of the models individually by giving the output with an accuracy rate of 96%.

As a modification to the project in the future, the dataset containing the features were downloaded from the internet from publicly available open source domain site called Kaggle. In-order to increase the efficiency, performance and accuracy of the model it is advised to extract the features manually rather than taking the dataset from the internet.

# 8 References


[1] Udemy. (n.d.). Android Malware Analysis - From Zero to Hero. [online] Available at: https://www.udemy.com/course/android-malware-analysis-from-zero-to-hero/ [Accessed 11 Nov. 2021].

[2] Tiwari, S.R. and Shukla, R.U. (2018). An Android Malware Detection Technique Based on Optimized Permissions and API. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/8597225 [Accessed 11 Nov. 2021].

[3] L. Sun, Z. Li, Q. Yan, W. Srisa-an and Y. Pan, "SigPID: significant permission identification for android malware detection", 2016 11th International Conference on Malicious and Unwanted Software (MALWARE), pp. 1-8, 2016

[4] X. Li, J. Liu, Y. Huo, R. Zhang and Y. Yao, "An Android malware detection method based on Android Manifest file", Cloud Computing and Intelligence Systems (CCIS) 2016 4th International Conference on, pp. 239-243, 2016, August.

[5] A. Arora and S.K. Peddoju, "Minimizing Network Traffic Features for Android Mobile Malware Detection", Proceedings of the 18th International Conference on Distributed Computing and Networking, pp. 32, 2017, January.

[6] K. Wang, T. Song and A. Liang, "Mmda: Metadata Based Malware Detection on Android", Computational Intelligence and Security (CIS) 2016 12th International Conference on, pp. 598-602, 2016.

[7] H.A. Alatwi, Android malware detection using category-based machine learning classifiers, Rochester Institute of Technology, 2016.

[8] L. Sayfullina, E. Eirola, D. Komashinsky, P. Palumbo and J. Karhunen, "Android Malware Detection: Building Useful Representations", Machine Learning and Applications (ICMLA) 2016 15th IEEE International Conference on, pp. 201-206, 2016, December.

[9] H. Zhang, Y. Cole, L. Ge, S. Wei, W. Yu, C. Lu, et al., "ScanMe mobile: a cloud-based Android malware analysis service", ACM SIGAPP Applied Computing Review, vol. 16, no. 1, pp. 36-49, 2016.

[10] Pei, W., Li, J., Li, H., Gao, H. and Wang, P. (2017). ASCAA: API-level security certification of android applications. IET Software, 11(2), pp.55–63

[11] A. S. K. Dash, "Mining API Calls and Permissions for Android Malware Detection," Cryptology and Network Security. CANS 2014. Lecture Notes in Computer Science, vol. 8813, pp. 191-205, 2014.

[12] Anon, (n.d.). Drebin: Effective and Explainable Detection of Android Malware in Your Pocket – NDSS Symposium. [online] Available at: https://www.ndss-symposium.org/ndss2014/programme/drebin-effective-and-explainable-detection-android-malware-your-pocket/ [Accessed 11 Nov. 2021].

[13] Yang, Y., Du, X., Yang, Z. and Liu, X. (2021). Android Malware Detection Based on Structural Features of the Function Call Graph. Electronics, 10(2), p.186..

[14] Zhang, W., Luktarhan, N., Ding, C. and Lu, B. (2021). Android Malware Detection Using TCN with Bytecode Image. Symmetry, 13(7), p.1107

[15] Darus, F.M., Ahmad, S.N.A. and Ariffin, A.F.M. (2018). Android Malware Detection Using Machine Learning on Image Patterns. 2018 Cyber Resilience Conference (CRC). [online] Available at: https://www.semanticscholar.org/paper/Android-Malware-Detection-Using-Machine-Learning-on-Darus-Ahmad/b035451da32748fd4e0997e00d3315e51e99d9e8 [Accessed 11 Nov. 2021].