National College of Ireland

Android Mobile Malware Detection System Using Ensemble Learning

MSc Research Project
Cybersecurity

Janish Ottakanchirathingal
Student ID: 19120788

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Janish Ottakanchirathingal
Student ID:	19120788
Programme:	Cybersecurity
Year:	2020
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	17/08/2020
Project Title:	Android Mobile Malware Detection System Using Ensemble Learning
Word Count:	5518
Page Count:	16

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.

I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	17th August 2020

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):	

Android Mobile Malware Detection System Using Ensemble Learning

Janish Ottakanchirathingal
19120788

Abstract

Android is the most popular and dominant smartphone operating system in the market. The Google Play store in the Android platform contains more than one million android mobile applications that are downloaded and used by users for various purposes. Meanwhile, they have also become a prime target of unethical intrusions due to their open-source platform and popularity. In this research, we observed that traditional methods failed to detect sophisticated malware as they are incapable of detecting and predicting malware with variable attributes. In the last many years, Machine learning classifications strategies have been used to tackle these issues and this research observes that ensemble learning can produce the best results. Thus, in this research, we proposed an Ensemble learning classification model using Decision tree with Gradient Boosting algorithm that was trained on the Malgenome mobile malware dataset. The performance of the model is evaluated using metrics like accuracy and F1 and the model's performance is compared with conventional models like Support Vector Machine (SVM) and Naive Bayes.

Keywords: Android, Malware, Ensemble Learning, Decision Tree with Gradient Boosting, SVM, Naive Bayes

1 Introduction

In the last two decades, the rapid development of high-speed mobile communication network has resulted in portable devices such as smartphones and tablets becoming more and more common. Individuals use smartphones for calling, texting, browsing the internet, listening to music, and carrying out various other activities. Android is a very popular platform provided by Google, that is powering billions of smartphones in the world. In Android smartphones, the Google play store contains all types of applications that could be downloaded and used by Android users. Increasingly human society depends on these applications to perform daily activities like to make payments, for booking a taxi, etc. As Android is an open-source platform, anyone is independent to develop an android based application. To perform various malicious activities, various intruders develop an Android application that contains the malware. It is important to note that intruders are well aware of the android popularity among the users that they have transformed their manual hacking procedures into the network-controlled bots that are adaptive to the purpose of data-stealing. The threat of android malware is evolving rapidly and increasingly users are faced with network-controlled ransomware, wiper malware, and the other adversaries that are used to break down the security measures.

1.1 Background Study

In 2019, Android's smartphone market share is a whopping 86.1%, and this huge popularity has made Android a favorite target of cybercriminals. The first Android malware, an SMS Trojan, was discovered in the wild by Kaspersky in the year 2010. Since then, Android malware has been evolving rapidly and incorporated sophisticated techniques to avoid detections. According to a study by Yajin Zhou and Xuxian Jiang [1], recent malware groups exhibit polymorphic behavior, malicious payload encryption, increased code obfuscation, stealthy command and control communications channels, dynamic runtime loading of malicious payload, etc. These anti-analysis techniques employed by the malware cause problems for traditional signature-based detection and significantly increase the effort in uncovering malicious behavior and code within the Android applications. Traditionally, there are two methods to detect malware applications, Static analysis, and dynamic analysis. In the static analysis, the application binaries are dissembled and then the application source code and related metadata is inspected without running the application. Whereas in dynamic analysis, the application behavior is inspected in the runtime.

As malware evolved and become more sophisticated, malware detection using these traditional approaches has become more difficult. In the last few years, Machine learning has produced critical results with their feature selection and malware classification approaches. Several supervised learning algorithms such as support vector machines (SVM), simple logistic, random forest, and a combination of classification and clustering were used to deal with the problem. In this study [2], they proposed the AntiMalDroid systems where the recognition of malware is conducted based on the behavior which uses the "logged behavior sequence" as the feature of "SVM model training and recognition". Thus, several techniques and methods using Machine learning algorithms are proposed by various researchers so that malware in the Android system could be detected effectively.

1.2 Research Rationale

Malware and their detection always encountered several issues as everyday new and emerging malware are introduced and developed by several hackers or attackers. Since the invention of computer systems, there has been malware targeting them for hacking and stealing confidential data. As the use of mobile and tablet is growing hugely, cybercriminals are increasingly targeting these devices. Android is the most dominant smartphone operating system in the market, and it is an open-source platform where anybody can develop an Android based application. It provides the flexibility of installing third party malicious applications to get confidential data from the users. As the Android system is modified by several developers, the attackers could modify the system files by clicking the security system of androids easily without letting the users know about it. As the users are also not much aware of malicious third-party applications, these issues are widespread and undetected. This research is responsible for detecting these types of malware affecting Android systems by using the ensemble learning to provide a safeguard to protect the Android system against unethical intrusions.

1.3 Research Aim

The research aims to analyze and gather knowledge about the malware detection systems effectively and understand the workflow of their detection by their behavior with the help of ensemble learning. The authentic secondary resources are responsible for gathering knowledge about the system and its infrastructures for analyzing these processes in terms of preventing malicious attacks and saving confidential data efficiently. The goal of this review is to emphasize how tree-based classifiers or ensemble classifiers are best for classifying the anomalies and malware in Android mobile phones and its usability.

1.4 Research Objectives

- To determine the effective work which can identify the new mobile application malware
- To determine the feature selection technique chi-square for achieving better accuracy.

1.5 Research Questions

- How accurately our proposed work can identify the new mobile application malware?
- How the feature selection techniques help us to achieve better accuracy?

1.6 Research Hypothesis

H1: The proposed work can identify the new mobile application malware very accurately.
H0: The proposed work cannot identify the new mobile application malware very accurately.
H2: The feature selection chi-square techniques help us to achieve better accuracy.
H0: The feature selection chi-square techniques do not help us to achieve better accuracy.

1.7 Conclusion

Ensemble learning was proposed in the year 1964 for classification purposes. Since then it has come a long way and it is well-liked and widely adopted among the machine learning community. By training multiple base learners and discriminating strong learners from weak learners, ensemble learning reduces error significantly. In this paper, we are focused on taking benefit of the advantages of the ensemble model by enhancing a traditional and efficient classification model: Decision Trees with gradient boosting. The work also focused on feature extraction and feature selection, and the model was then trained and tested. Its performance is evaluated using metrics like accuracy and F1 score. The model's performance is then compared with traditional models like SVM and Naive Bayes to see if the ensemble models outperform traditional models.

The remaining sections of the research paper are as follows: Section 2 discusses the related work, Methodology carried out for the research is described in Section 3. The design specification is described in Section 4, while Section 5 covers implementation.

Section 6 covers the Evaluation of the results, followed by a conclusion and future work in Section 7.

2 Related Work

Human society depends on technology and the internet now more than ever to carry out their daily activities. The adoption of the internet has rapidly risen in the last decade largely thanks to smartphones. With the introduction of the first Android phone in 2008, the use of mobile phones has been exploded since then. Besides using mobile phones for audio and video calls, people also use Android applications downloaded from Google Play store for various purposes. The rise in popularity of android phones has gained the attention of cybercriminals and threat actors. This has led to the plethora of mobile malware applications targeting mobile phones running Android operating system to conduct nefarious activities such as stealing money or sensitive information, IP theft, extortion, and more. These malware applications are designed by the creative hackers that require intelligent processing and detection agents to detect.

Traditional signature-based detection models that worked in the last many decades have become inadequate in this age as they simply cannot cope up with the sheer number of malware application developed daily. But from the last several years, Machine learning has produced critical results with their feature selection and malware classification approaches. This work will review the important classification methods that were used to detect the anomalies in the mobile application domain together with the brief background of some traditional approaches. The goal of this review is to emphasize how tree-based classifiers or ensemble classifiers are best for classifying the anomalies and malware in mobile phones and their usability.

2.1 Empirical Study

One of the biggest inventions in terms of technology is mobile phones. In the early days, people use mobile phones for making calls, sending text or multimedia messages. With the invention of smartphones, people use mobile phones for surfing the internet, listening to music, downloading, and sharing purposes besides calling and sending messages. The evolution in the use of these mobile phones can be seen from the initial days to the present day that there is a huge change that has occurred throughout these days. The Google Play store is the repository in the Android platform from where Android applications can be downloaded. People use these Android applications to fulfill various purposes. It is also a common trend that people pair their Android phones with different web applications or with other individuals at the public places that will increase the probability of data-stealing through interconnected mobile malware applications. These malware applications are designed by the creative hackers that require intelligent processing and detection agents to detect.

According to Suleiman, the malware is nothing but a malicious program that gets injected in the different process by the cybercriminals, and without the knowledge of the owner, the criminals enter into the system and cause damage by getting personal or sensitive data from the system and use it for nefarious purposes [3]. There are different approaches to minimize the risk and to keep the cybercriminals away from the

system. In this research, the concept of "Ensemble Learning Algorithm" will be taken into consideration to incorporate with the Android phone so that it can detect any malware activities that might be injected by any unwanted sources. If the malware or different viruses get identified by this algorithm, then it will be quite easier for the developer of the Android phones to incorporate this algorithm into the system so that users will not face any sort of unwanted activities while using their phones. The malware researchers also use static analysis and dynamic analysis to detect malware. A Static analysis of software is performed without executing the program. Static analysis technique based on system calls, source code, and taint is shown in figure 1. Dynamic analysis requires that we execute the program, often in a virtual environment.

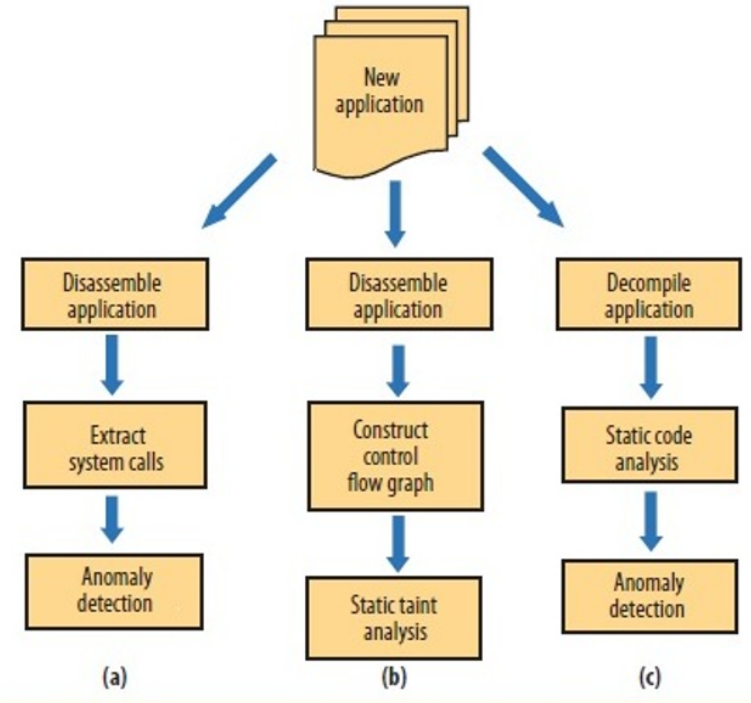

Figure 1: Static Analysis based on (a)system calls, (b) taint, (c) source code

In the early days, Antivirus software was used to confront Android mobile malware. Android phones were equipped with an antivirus scan tool that can detect malware functions in smartphones. Before incorporating the software, it needs to make sure that it runs of complete scan and does not overlook any sort of vulnerabilities [4]. According to Sakir, we must keep the software updated with new malware signatures, so that it can recognize the present version of the malware and keep our Android device safe.

Traditional defense techniques such as Antivirus became inefficient in recent times, as they simply could not keep up with the pace of development and introduction of continuously updated malware. Moreover, as they rely on signature-based detection, they were incapable of defending against zero-day attacks or to detect a new malware. The antivirus software also failed to stop short message service (SMS) scams that are used to steal the user data [1]. According to Oberheide, it almost took around 48 days for an antivirus scanning tool to detect new threats that can enter the system [5]. In this study [6], the author reports that the use of network-controlled bots than human-controlled bots to target mobile phones are highly appreciated among the cybercriminals

and black hat hackers. Then, IDS systems were developed and unlike Antivirus software, IDS has proved to be vital in malware detection with their anomaly and misuse-based IDSs [7]. While misuse-based IDS were not successful as they failed to detect unknown malware, the anomaly-based IDS were widely used since they detect malware based on anomalous activity. But in their research [8], the author claims that with the evolution of malware and increased sophistication techniques these IDS systems are of very little use. Hence, they suggest that the Machine Learning strategies are excitingly good because they predict the malware according to the labeled or unlabeled data sets and provide stable detection strategies with excellent model accuracies.

In the last many years, Machine learning has been used to classify malware targeting computers, servers, and computer systems. In recent years, with the availability of more samples of Android malware, Machine learning is increasingly used to detect malicious applications. In this research, the concept of the "Ensemble Learning Algorithm" was taken into consideration to incorporate with the Android phone so that it can detect any malware activities that might be injected by any unwanted sources. The concept of Ensemble Learning is nothing, but the way of diversification based on the decision of machine learning is called ensemble [9]. It can be said that the decision of an individual cannot be more appropriate compared to a group of feedback or rating such as if someone wanted to buy a car, that person must consult an individual and get a single positive or negative response. Still, if that person asks a group of people to make a choice based on a group of cars, then they will come up with a conclusion after a certain discussion by seeing the negative and positive side of the cars. The exact similar things happen in machine learning where the diversification of results is considered for taking few certain decisions. It also incorporates machine learning but grouping a bunch of diverse decisions and come to one single result or conclusion is the main responsibility of this algorithm technique [10]. The method of algorithm combines with different techniques of machine learning into the single predictive model to decrease bias, Variance, and make betterment to the prediction. These terms can be technically called boosting, bagging, and stacking, respectively.

The role of ensemble learning in detecting the android malware is of prime importance because the ensemble learner combines the uniqueness of different ML techniques as a multilevel or tree-based classifier to eliminate the learning loss and increase the variable immunity of the classification system. The main reason behind using machine learning algorithms in the Android system for malicious behavior detection of the applications is due to the intrinsic nature of malware that it can remain quite a long time without getting notified. Most of the traditional approaches for preventing malware are based on keeping the malware signature updated in the detection system. So, the use of antivirus will not be helpful to detect any existing malware which is there for a long time in our system but hidden as its signature has not yet been identified as malware or any new unencountered malware. Whereas when we use machine learning algorithms, their nature is to learn from their experience and as models are based on certain features, so they will help a great deal when we put it to use in the proposed system where even without updates we can uncover any new malware which we thought as a safe application.

In this study [11], the authors applied Bayesian classification to group the applications into benign or suspicious. They used 1000 samples of each benign and malware application for the training and classification. Whereas in this research [12], authors used permission and call flow graphs for training SVM models to classify benign and malicious applications. In a research by Aravind and Paramvir [13], they tried to classify malicious

android applications using Naïve Bayes, Random Forest, J48 Decision Tress, K-star, and Simple logistic technique. This work concludes that simple logistic technique has achieved an accuracy of 99.7 percent on the dynamic application dataset the choice of classifier and the less data sample often compromise the detection precision and resulted in false positives. A behavior-based malware detection system by Iker and Simin [14] use k-means algorithms on system calls behavior to detect malware. Hemalatha and Selvabrunda in their research [15] used mixed-kernel support vector machine (SVM) model to detect the android malware with the use of metrics that support precision issues and achieved an accuracy of 96.89 percent. In this reported research [16], the author claims that network traffic analysis and modelling of the imbalanced network traffic is important for the feature selection, boosting the immunity of the detector and the resulting accuracy of the ML classifier. A signature-based malware detection approach using Machine learning algorithms is shown in figure 2 [17].

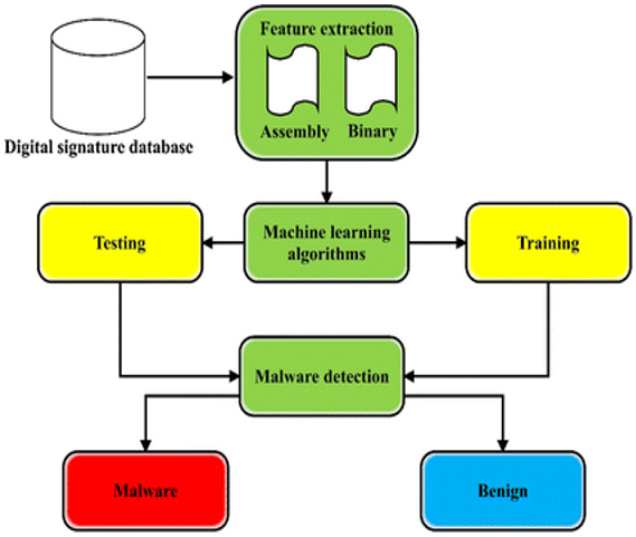

Figure 2: Malware detection approach

According to many researchers, the use of multi-level classifiers increases model accuracy over single classification algorithm. In the recent past, the concept of decision trees (multi-level classifiers) together with ensemble learning was introduced that increases the model efficiency and accuracy. An example of this concept was studied in this research [18], where they uses the decision trees, random forest, and gradient boosting to analyze the multi-hierarchy classification that gave 97.92 percent of accuracy in the mobile malware detection. Another multi-level ensemble detector is reported in this research paper [19], where they have developed the multi-level classifiers (Droid Fusion) i.e. the base classifiers at the lower stage and the predictive ranking-based algorithm at the higher-level to increase the classifier accuracy by eliminating the security threats to the android system. DroidMat [20], where Android malware is detected using k-means clustering after computing the required number of clusters by Singular Value Decomposition (SVD) approach. They present experimental results based on 238 Android samples from 34 families together with 1500 benign apps. The evaluation of the different ensemble or tree-based classifiers were discussed in the research by Sohel Rana and Andrew H Sung, where they have used the Random Forest [21]. SVM, Extremely Randomized Tree, and

Gradient boosting tree together with the substring-based methods are used to eliminate the improper and unwanted information from the dataset, thus reduce the feature size to avoid the larger data dimensions. The authors also argued that on some data set and metric variation Gradient boosting provides the best result, but it consumes a lot of training time. As our research deals with gradient boosting tree for malware detection we took this point in our research that how to reduce the training time when applying the Gradient boosting ensemble classification. In a study Fauzia [22], to detect the malware ensembles were deployed with decision trees i.e. stacking, boosting, and bagging. The author argues that the gradient boosting will perform better if the weights of the boosting algorithm is treated as non-heuristic. In the research by Kichang Kim, they implement the tree boosting ensemble learner to predict the malicious application by calculating the risk score based on the code evaluation strategies that improve the performance of the learner [23].

Various traditional methods have been used, which uses the blacklisting approach to detect the malware. Since they were unable to detect new malware these approaches were of little use. The research papers studied for this dissertation all cover various techniques like static analysis, dynamic analysis, and Machine learning approach.The Static analysis fails at sophisticated code obfuscation used by the attacker and also at polymorphic and metamorphic malware. Dynamic analysis needs a secure isolated environment and the malware may behave differently in the two environments (virtual and real), besides, some actions of malware are triggered under some specific conditions like (data, time, platform) in the real environment which may not be detected by the secure virtual environment. Using machine learning, we can overcome these drawbacks as the models are based on features. When it predicts the malware or benign nature of application based on the features that are significant, and with proper training of the models, it may even uncover the malware that is new or are still hidden in plain sight. In this work, we proposed the decision tree with gradient boosting strategy that will consider non-heuristic learning weights optimization, selection of the best features through network feature dataset, reduce the classifier training time with the autonomous elimination of the unwanted tree extensions, and the fast boosting algorithm that solves the multi-classification problem instead of binary classification and yields the best classification accuracy.

3 Research Methodology

The process to build the Android malware detection system will be done in a series of steps as shown in figure 3. First, the dataset that we will be working on will be taken to the system. It will be followed by data preprocessing. In this step, understanding of the data is established. The data is checked for missing values, outliers, etc if found any of these issues, the necessary preventive measures are performed. Then feature extraction and selection are performed. In this stage, we are using chi-square as a measure to score the features based on the significance level in reaching to the target variable using that feature. The score is then compared and the one's with the lesser score is dropped out of the data. Then the proposed models are implemented. The models which will be implemented are decision tree with Gradient Descent discussed in section 4.1, Support Vector Machine (SVM) discussed in section 4.2, and Naive Bayes in section 4.3, The evaluation metrics discussed in section 4.4 are accuracy score and F1 score.

The proposed models for a malware prediction system put together in this paper are

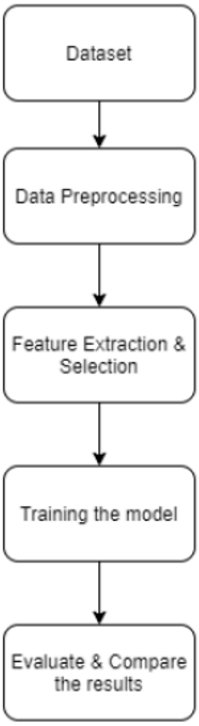

Figure 3: Proposed Model

Decision trees with gradient descent, SVM, and Naive Bayes. All these models fall under the category of classification models. Classification models work with the data to identify which class or category an observation belongs too. The first step is to train the models on the training data, followed by testing on the test data. Training and test are nothing but the subsets of the entire population. The training set is used to train the model to learn and test the model for its performance is done using the test data. The values in actual data and one predicted by the model are compared for the accuracy score is generated.

4 Design Specification

The machine learning models and evaluation metrics that are implemented in this proposed system are described below and the conceptual framework is shown in Figure 4 :

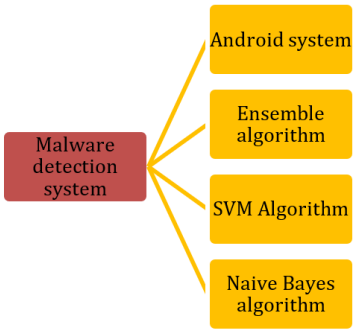

Figure 4: Conceptual Framework

4.1 Decision Trees with Gradient Boosting

For ensemble we need a base algorithm, a final algorithm is added as a function on the base algorithm. The model is improvised by implementing step learning. The ensemble techniques which are popular now-days are boosting and bagging. Boosting is adding a weight to the classifier and bagging is taking prediction average over a classifier group. To generate a decision tree ensemble, we are using gradient boosting. In decision tree, we implement a step that is weight, a random value. We run the model through various step sizes, to find the optimal step size. As in this method, a larger value of step-size will make us skip the local minima.

4.2 Support Vector Machine (SVM)

Support Vector Machines (SVM) shows the data points in training data in the space that separates the category by a distinct gap known as a hyperplane. In training SVM model, the data points are put in their respective category group in the space. The hyperplanes are then generated to mark the distinct groups. It is used for both regression and classification purposes. SVM adds a margin in the hyperplanes, which is the distance between the nearest data point in a class.

4.3 Naive Bayes

Naïve Bayes is a probabilistic classifier based on Bayes theorem which assumes that there is a strong relationship within the features. It forms a simple Bayesian network model coupled with Kernel density estimation to achieve high accuracy.

4.4 Proposed Evaluation Metrics

For a machine learning model, the evaluation metrics are an integral part. There is no point in creating a machine learning model that does not have a well-defined feedback mechanism. For a model, we receive its feedback from the evaluation metrics; we keep on improving the model performance until we reach the desired accuracy score. Evaluation metrics guide the performance of a model. In the proposed system, the performance of the models are evaluated using the following parameters:

4.4.1 Accuracy

Accuracy is the percentage of the sum of all the correct predictions with the total number of observations. The accuracy score is a statistical measurement scale for a model. Accuracy is the number of values that the model can predict efficiently. Mathematically it can be represented as:

Accuracy = (True positive + True negative /(True positive + True negative + = False-positive + False negative)

True Positive is the number of actually true values, and the model also predicted to be true. True negative is the number of actually negative values, and the model also predicted to be negative. False Positive is the number of actually negative values, and the model also predicted to be true. False-negative is the number of actually true values, and the model also predicted to be negative.

4.4.2 F1-Score

F1-Score is the harmonic mean of precision and recall values for a classification problem. F1 Score is mathematically represented as:

For the F1 score, the harmonic mean is taken into consideration as it punishes extreme values more. F1 Score is within the range [0, 1]. There has to be a proper balance between precision and recall values. As with a high value of precision with a low value of recall, provides us with an accurate model, on the contrary leaving values that are out of the classification range. For an efficient model, the value of F1-Score is on the higher end. Precision is closeness to the actual values, calculated by using the below formula:

Precision= True Positive / True Positive + False Positive

The recall is the ability of the model to find the relevant values.

Recall= True Positive / (True Positive + False negative)

True Positive is the number of actually actual values, and the model also predicted to be true. The real negative is the number of actually negative values, and the model also predicted to be negative. False Positive is the number of actually negative values, and the model also predicted to be true. False-negative is the number of indeed actual costs, and the model also predicted to be negative.

5 Implementation

In this section, we describe how the implementation was carried out to build an efficient classification model using ensemble learning. The modelling and feature selection process is also described in this section.

5.1 Environment Setup:

The proposed model is coded in the python programming language. The python distribution widely used for data science, Anaconda software, and Jupyter Notebook (IDE) is used for writing and executing the code on Microsoft Windows 10 platform.

5.2 Dataset Description

The Malgenome mobile malware dataset provided by North Carolina State University is used in this research for training and testing the model. The publicly available dataset was downloaded from figshare [24] in CSV format. The dataset consists of 215 mobile application features with the 3799 samples (1260 malware apps from Android malgenome project and 2539 benign apps). The dataset feature involves trsact, onService-Connected, ServiceConnection and send SMS etc. Which will be used for our analysis.

5.3 Package Installation

The following required libraries and packages are installed to perform our research:

1. **Numpy:** Numerical Python or Numpy library is used for array related operations
2. **Pandas:** Pandas Library in python is used for data manipulation and analysis
3. **Matplotlib:** Matplotlib library in python is used for plotting the graphs
4. **Scikit Learn:** Scikit Learn library is the Machine learning package of python programming language. It contains optimized code for various machine learning algorithms like the random forest, logistic regression, naïve bayes, etc.

5.4 Exploratory Data Analysis

Once the required packages are installed, data is imported into the system, we perform data pre-processing where a basic statistical description of all the features is done, where the count, mean, mode, standard deviation, and quartiles are calculated as shown in Fig 5.

	transact	bindService	onServiceConnected	ServiceConnection	android.os.Binder	READ_SMS	attachInterface	WRITE_SMS
count	3799.000000	3799.000000	3799.000000	3799.000000	3799.000000	3799.000000	3799.000000	3799.000000
mean	0.452224	0.467228	0.470387	0.469334	0.500921	0.256383	0.437484	0.202948
std	0.497778	0.498991	0.499188	0.499124	0.500065	0.436693	0.496142	0.402247
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Figure 5: Statistical description of all features

Then the data is checked for missing values and this dataset does not contain any null values or missing values.

5.5 Feature Selection

One of the most important steps in a Machine learning classification is selecting significant features for training our model. Feature selection is the process of selecting the correct number of features to train our model and ignore insignificant feature to increase the precision and to reduce training time. The data features and target set are separated and then features are evaluated for their level of significance using chi- square method. There was a total of 216 application features in the dataset. It was discovered that by using chi-square that out of 216 application features only 98 are significant.

5.6 Modelling

The objective of this research was to build a model to classify Android malware application into malware or benign using ensemble learning. We have used Decision tree with gradient descent algorithm as an ensemble learning model and support vector machine (SVM) and Naïve Bayes algorithms are used for comparative study. After the feature selection using Chi-Square method, the data is then checked for overfitting or underfitting. By overfitting, it is meant that the features that are fed into model are too much for the model to handle. Underfitting is when the number of features in the model are too few. Both affect the performance and the accuracy of the model. The models are then built and tested for accuracy and f1 score.

6 Evaluation

The models are implemented, and their performance is evaluated, it is found the Gradient Boosted Decision Tree outperforms the other two models i.e. SVM and naive bayes.

Model	Accuracy	F1-Score
GBDT	99.12	99.12
SVM	99.04	99.04
Naive	99.04	99.04

Figure 6: Model Scores

The scores for the implementation are shown in figure 6 above. From the figure, we can see that the best scores belong to Gradinet boosed Decision Tree or GBDT . Gradient boosted Decision Tree has an accuracy score of 99.12% and F1 score of 99.12, Support Vector Machine has an accuracy of 99.04% and F1 score of 99.04 and the Naive Bayes has the same scores.

7 Conclusion and Future Work

The rise in the adoption of Android smartphones continues to grow and this is why Android malware is such a lucrative business for bad actors. As we observed in this research, the traditional security measures are not efficient in mitigating the risk of intrusion due to malware. As attackers use sophisticated techniques and measures, there is a dire need for an efficient safeguard to protect Android systems from illegal intrusions. The aim of this research was to find the ensemble model performance in predicting the existence of malware in the android system. The Decision tree with gradient boosting algorithm was trained on the dataset and the ensemble model had an accuracy of 99.12 in predicting if an application is malware or benign. It was found that the ensemble model outperforms the Support Vector Machine and Naive Bayes by a slight margin. So, with this, we may conclude that all the models are good at predicting if a particular application is benign or malware and our ensemble model have higher success rate than SVM and Naïve bayes models using the same dataset.

For future work, we can compare the various ensemble models like AdaBoost, XGBoost and check which one is best for the said purpose. We can also apply neural networks and see the difference in the performance.

References

[1] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *2012 IEEE symposium on security and privacy*. IEEE, 2012, pp. 95–109.

[2] M. Zhao, F. Ge, T. Zhang, and Z. Yuan, "Antimaldroid: An efficient svm-based malware detection framework for android," in *International Conference on Information Computing and Applications*. Springer, 2011, pp. 158–166.

[3] S. Yerima, S. Sezer, and I. Muttik, "High accuracy android malware detection using ensemble learning. iet inf. secur. 9 (6), 313–320 (2015)," 2014.

[4] F. Alswaina and K. Elleithy, "Android malware permission-based multi-class classification using extremely randomized trees," *IEEE Access*, vol. 6, pp. 76 217–76 227, 2018.

[5] J. Oberheide, E. Cooke, and F. Jahanian, "Cloudav: N-version antivirus in the network cloud." in *USENIX Security Symposium*, 2008, pp. 91–106.

[6] S. Kothari, "Real time analysis of android applications by calculating risk factor to identify botnet attack," in *ICCCE 2019*. Springer, 2020, pp. 55–62.

[7] S. Shamshirband, N. B. Anuar, M. L. M. Kiah, and A. Patel, "An appraisal and design of a multi-agent system based cooperative wireless intrusion detection computational intelligence technique," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 9, pp. 2105–2127, 2013.

[8] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Computing*, vol. 20, no. 1, pp. 343–357, 2016.

[9] P. Feng, J. Ma, C. Sun, X. Xu, and Y. Ma, "A novel dynamic android malware detection system with ensemble learning," *IEEE Access*, vol. 6, pp. 30 996–31 011, 2018.

[10] L. Vaishanav, S. C. H. Vaishanav, and M. S. S. D. R. Kumar, "Behavioural analysis of android malware using machine learning," *International Journal of Engineering and Computer Science*, vol. 6, no. 5, 2017.

[11] S. Y. Yerima, S. Sezer, G. McWilliams, and I. Muttik, "A new android malware detection approach using bayesian classification," in *2013 IEEE 27th international conference on advanced information networking and applications (AINA)*. IEEE, 2013, pp. 121–128.

[12] J. Sahs and L. Khan, "A machine learning approach to android malware detection," in *2012 European Intelligence and Security Informatics Conference*. IEEE, 2012, pp. 141–147.

[13] A. Mahindru and P. Singh, "Dynamic permissions based android malware detection using machine learning techniques," in *Proceedings of the 10th innovations in software engineering conference*, 2017, pp. 202–210.

[14] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, 2011, pp. 15–26.

[15] H. I. A. and Selvabrunda, "Mobile malware detection using anomaly based machine learning classifier techniques," in *Proceedings of the International Journal of Innovative Technology and Exploring Engineering*, vol. 8, 2019, pp. 260–267.

[16] Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, and B. Yang, "Machine learning based mobile malware detection using highly imbalanced network traffic," *Information Sciences*, vol. 433, pp. 346–364, 2018.

[17] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Computing and Information Sciences*, vol. 8, no. 1, p. 3, 2018.

[18] M. S. Rana, C. Gudla, and A. H. Sung, "Android malware detection using stacked generalization," in *27th International Conference on Software Engineering and Data Engineering*, 2018.

[19] S. Y. Yerima and S. Sezer, "Droidfusion: a novel multilevel classifier fusion approach for android malware detection," *IEEE transactions on cybernetics*, vol. 49, no. 2, pp. 453–466, 2018.

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

[21] M. S. Rana, S. S. M. M. Rahman, and A. H. Sung, "Evaluation of tree based machine learning classifiers for android malware detection," in *International Conference on Computational Collective Intelligence*. Springer, 2018, pp. 377–385.

[22] F. Idrees, M. Rajarajan, M. Conti, T. M. Chen, and Y. Rahulamathavan, "Pindroid: A novel android malware detection system using ensemble learning methods," *Computers & Security*, vol. 68, pp. 36–46, 2017.

[23] K. Kim, J. Kim, E. Ko, and J. H. Yi, "Risk assessment scheme for mobile applications based on tree boosting," *IEEE Access*, vol. 8, pp. 48 503–48 514, 2020.

[24] S. Yerima, "Android malware dataset for machine learning 1," https://figshare.com/articles/Android_malware_dataset_for_machine_learning_1/5854590/1.