COMPARISON OF MFCC, BFCC, and NGCC IN ANDROID MALWARE DETECTION

**HONOURS**

**RESEARCH DISSERTATION BY**

|  |  |
| --- | --- |
| **Lungelo Mkhize** | **Researcher** |
| **Mr.P Tarwireyi** | **Research Supervisor** |

**Department of Computer Science**

A picture containing text, clipart

Description automatically generated

**University of Zululand**

DECLARATION

I, Lungelo Mkhize, hereby declare that I am the only author of this mini-dissertation, and it has never been submitted, either in whole or equivalent another application for a degree. The work offered is completely my own, unless stated otherwise by reference or acknowledgment.

30 November 2022

Lungelo Mkhize Date

**ACKNOWLEDGEMENTS**

I would like to thank my one and only supervisor Mr. Paul Tarwireyi who has been my supervisor since I started this paper. Thanks to him for requiring more than I thought was expected of me. Through him, I am confident that I can achieve far with the research skills that I have gained from him. and I see no limit to my capabilities. I also want to acknowledge the Drebin website for giving me the dataset I used to complete my work with a successful result.

In honor of the people, I remember from the Computer Science department, without whom I wouldn’t have all the knowledge I have today, I pay my deep thanks for their dedication to me and all my colleagues, from when we had no idea what computers were, to now, when we understand the science behind them.

I cannot forget to acknowledge all my fellow computer science Honours classmates. It was an honor working with you, working through almost every problem together, and facing similar challenges. Thank you for all that you did for yourselves in front of me, you encouraged me to do the same for myself.

**ABSTRACT**

The Android operating system has always been in the spotlight. Cybercriminals have recognized attracting the growing challenges. However, many researchers have achieved success by employing machine/deep learning techniques to build malware detection models based on popular Drebin malware datasets. The paper presents an audio feature malware detection system, using machine learning classifiers for the Drebin dataset (APK) which is treated as acoustic signals and we used audio feature extraction techniques named Normalized Gammachirp Cepstral Coefficients (NGCC), Bark Frequency cepstral coefficients (BFCC) and Mel-frequency cepstral coefficients (MFCC). Based on the results obtained from the experiment, BFCC is the best audio feature compared to other authors’ works. The BFCC audio feature has a high accuracy score of 99.0% with a precision of 99.0%, F1-Score of 99.0%, and recall of 99.0% compared to other authors although NGCC and MFCC results are encouraging but (CHENGLIN LI et al., 2019) has the better results than them.

***Keywords****: android malware detection, Mel Frequency cepstral Coefficients, Bark frequency cepstral coefficients, and Normalized Gammachirp Cepstral Coefficients.*

CONTENTS

Contents

[List of figures and tables 5](#_Toc120635793)

[1 Chapter 1 – INTRODUCTION 6](#_Toc120635794)

[1.1 Problem Statement 8](#_Toc120635795)

[1.2 Research Questions 8](#_Toc120635796)

[1.3 Rationale 8](#_Toc120635797)

[1.4 Research Goals and Objectives 9](#_Toc120635798)

[1.4.1 Research Goal 9](#_Toc120635799)

[1.4.2 Research Objectives 9](#_Toc120635800)

[2 Chapter 2 – LITERATURE REVIEW 10](#_Toc120635801)

[3 Chapter 3 - RESEARCH METHODOLOGY 14](#_Toc120635802)

[3.1 Data collection 14](#_Toc120635803)

[3.2 Feature Engineering 14](#_Toc120635804)

[3.2.1 Normalized Gammachirp Cepstral Coefficients (NGCC) 15](#_Toc120635805)

[3.2.2 Bark Frequency cepstral coefficient (BFCC) 16](#_Toc120635806)

[3.2.3 Mel Frequency Cepstral Coefficients (MFCC) 17](#_Toc120635807)

[3.3 Machine learning algorithms evaluation 18](#_Toc120635808)

[3.4 Classification Metrics 21](#_Toc120635809)

[3.4.1 Accuracy - is obtained by dividing the total number of predictions by the true positive predictions. 21](#_Toc120635810)

[3.4.2 Precision - Precision is defined as the ratio of true positives to total positives predicted. 21](#_Toc120635811)

[3.4.3 Recall - is the ratio of true positives to all the positives in dataset. 21](#_Toc120635812)

[3.4.4 F1-Score - It combines accuracy and recall and is the harmonic mean of the two. 22](#_Toc120635813)

[3.4.5 Area under the curve - uses both false positive and true positive rates. 22](#_Toc120635814)

[4 Chapter 4 - RESULTS AND DISCUSSION 23](#_Toc120635815)

[4.1 Confusion Matrix 23](#_Toc120635816)

[4.2 Classification performance 25](#_Toc120635817)

[4.3 Comparison of the study with existing literatures 27](#_Toc120635818)

[4.4 Comparison graphs 29](#_Toc120635819)

[4.5 Precision-Recall Graphs 32](#_Toc120635820)

[5 Chapter 5 – Conclusion 34](#_Toc120635821)

[6 REFERENCES 35](#_Toc120635822)

## List of Figures and Tables

[Figure 1: Number of Android malware per year(AV-TEST Institute, 2021) 7](#_Toc120637150)

[Figure 2: Proposed methodology architecture diagram( Mercaldo. F & Santone. A, 2021) 15](#_Toc120637151)

[Figure 3: Flowchart of NGCC Feature Extraction(K. Ouni, 2016) 16](#_Toc120637152)

[Figure 4: Flowchart BFCC Feature Extraction(K. Ouni, 2017) 17](#_Toc120637153)

[Figure 5: Flowchart of MFCC Feature Extraction( Magre S.B & Janse P.V, 2014) 17](#_Toc120637154)

[Figure 6: BFCC Confusion Metrics 23](#_Toc120637155)

[Figure 7: NGCC Confusion Metrics 24](#_Toc120637156)

[Figure 8: MFCC Confusion Metrics 24](#_Toc120637157)

[Figure 9: NGCC graphs for Accuracy and AUC 29](#_Toc120637158)

[Figure 10: MFCC graphs for Accuracy and AUC 30](#_Toc120637159)

[Figure 11: BFCC graphs for Accuracy and AUC 31](#_Toc120637160)

[Figure 12: BFCC precision-recall 32](file:///E:\final\Final%20Dissertation.docx#_Toc120637161)

[Figure 13: BFCC precision-recall 32](file:///E:\final\Final%20Dissertation.docx#_Toc120637162)

[Figure 14: MFCC Precision-recall 32](#_Toc120637163)

[Figure 15: NGCC precision-recall 33](#_Toc120637164)

[Table 1:Dataset used for this study 16](#_Toc119997590)

[Table 2:BFCC models performance 17](#_Toc119997591)

[Table 3: NGCC models performance 18](#_Toc119997592)

[Table 4: MFCC Models performance 19](#_Toc119997593)

[Table 5: Comparison of the study and other authors 20](#_Toc119997594)

List of Equations

[Equation 1: Calculate NGCC ( Y. Zouhir & K. Ounis, 2017) 15](file:///E:\final\Final%20Dissertation.docx#_Toc120637608)

[Equation 2: NGCC feature extraction ( Y.Zouhir & K. Ouni, 2016) 15](file:///E:\final\Final%20Dissertation.docx#_Toc120637609)

[Equation 3: Calculate BFCC (P. Tarwireyi, 2022) 16](#_Toc120637610)

[Equation 4: F1 score 22](#_Toc120637611)

[Equation 5: AUC 22](#_Toc120637612)

[Equation 6: Precision- recall 32](#_Toc120637613)

# Chapter 1 – INTRODUCTION

Android is the operating system that has gained significant popularity in the mobile telecommunications industry due to its large user base.According to 2020 statistical data, there are approximately 3.5 billion mobile device users around the entire world (Conker, 2020). The increase in the development of android devices has become the major attack target of malware. Malware can be a malicious file or code deployed to the system to perform the task the attacker intended. Since there are variants of malware, there are a lot of ways to deploy android malware into the system. These ways include downloading malicious apps, opening suspicious emails, receiving text messages/voicemail phishing, etc. (Arslan RS,2021).

Smartphones have become a crucial part of our everyday lives; we almost exclusively rely on them for the critical data we save on them. However, the flexibility of android smartphones has allowed attackers to take advantage of them. It is possible for hackers to install their malicious application onto android phones since android devices enable third parties to install the program into their system. When an application is installed, the user is prompted to approve all permissions. Cybercriminals typically used this as an advantage to request the user's full consent before accessing confidential data. When installing the application, the user consents to all permissions without realizing that they are being requested, necessitating the necessity to identify Android malware before it infiltrates the device.

Several alternative ways for detecting malware on Android have been offered by researchers. The techniques of malware detection that have been employed may be divided into three groups: static, dynamic, and hybrid analysis based. Although it is challenging to evaluate binary code due to the proliferation of obfuscation techniques, such as encryption and data masking, etc., static analysis looks at the binary code and attempts to identify any code that has not yet been executed. Since static analysis relies on a predefined signature database, it is unable to identify fresh, undiscovered malware. Malware may be executed, and its activities tracked using the dynamic analysis approach, which has proved successful in detecting malware. Both static and dynamic approaches detect malware by comparing unusual and predictable behaviour.

More than 50 million instances of malware and potentially unwanted applications (PUA) for Android have been discovered, according to data (AV-TEST institute, 2021).

Chart, bar chart

Description automatically generated

Figure 1: Number of Android malware per year (AV-TEST Institute, 2021)

By combining both static and dynamic analysis to extract aspects of system processes, machine learning has been suggested by previous authors as a method for detecting Android malware. Malware detection often makes use of machine learning algorithms like Naive Bayes (NB), Decision Trees, Support Vector Machines and K-Nearest Neighbor (KNN), (SVM) (Ahmed S. 9 May 2022). Despite certain challenges, algorithms can nevertheless produce useful behavior to enhance malware detection performance. The case for using traditional machine learning techniques to identify malware has not yet been made.

Many researchers have examined the effectiveness of the models in the Drebin dataset by applying machine learning techniques for the identification of android malware. In this paper, we used the same dataset to develop the audio feature malware detection system and the dataset is treated in the format of WAV. We applied audio feature extraction techniques, named as Bark Frequency cepstral coefficient (BFCC), Normalized Gammachirp Cepstral Coefficients (NGCC), and Mel Frequency Cepstral Coefficients (MFCC**).** When users use the ASR system on their smartphones to retrieve information while being interrupted by other people's conversations in a vehicle or subway system (Ta-Wen Kuan, 2015), for example, the accuracy performance of ASR is evaluated using the BFCC approach. However, although MFCC has been investigated for its performance in speech recognition systems, the Fourier transform and the triangular Mel-filter bank used in MFCC have been shown not to be representative of the sound wave sensitivity at the basilar membrane in the human auditory system, as well as having lower robustness in the presence of additive noise (Ta-Wen Kuan, 2015).

We tested our model using two datasets of Android malware: DREBIN (Daniel Arp et al, 2014) and the Google Play Store dataset that was directly downloaded, which each included 5,560 samples (APK). We used the key metrics to evaluate performance: accuracy, false positive rate (FPR), precision, recall, F1, and training time.

## Problem Statement

Our daily lives have become more and more dependent on android devices. This is because most people use them for learning and research, online banking, finance, as well as voice, message, and email communication. The dominance of this operating system has made it the major target for cyber attackers. Cyber attackers deploy malware on android devices to delete, modify and steal critical information. However, Static and Dynamic malware detection approach has been introduced to solve this issue. However, they are not effective nowadays because cybercriminals have studied these strategies and developed updated variants of malwares. It has become difficult to develop new techniques to combat new malwares. Although dynamic has executed and analyzed the malware threat and its behaviour. In this study we focus on audio-based features in malware detection.

## Research Questions

In this study, we consider the following research questions to develop an audio feature malware detection system:

1. What is the state of the art regarding malware detection?
2. How can we design a malware detection system using audio features?
3. How do BFCC, MFCC and NGCC compare in Android malware detection?

## Rationale

Since people rely on the android operating system for their critical information, it still remains a major target for cyber attackers. Because cyber attackers see the opportunity to install malware on this operating system to delete, modify, and steal data. Since people still lose their sensitive information, there is still a huge gap in android malware research. In this paper, we will use the Drebin dataset and treat it as an audio signal to perform feature extraction. This paper is based on comparing the audio-based features in malware detection with other authors ‘work to determine whether they are excellent and if so, which feature is the most effective.

## Research Goals and Objectives

### Research Goal

The aim of this study is to compare BFCC, MFCC and NGCC results to develop audio malware detection system.

### Research Objectives

1. To determine the state of the art regarding malware detection.
2. To compare BFCC, MFCC, and NGCC features with other existing studies.
3. To provide the best audio feature for malware detection.

# Chapter 2 – LITERATURE REVIEW

## Malware Analysis Techniques

After a thorough review of the research in the field of Android malware detection. Android malware has become a security threat on mobile devices. I saw a variety of study objectives. Many research papers look at present solutions to dealing with the malware detection problem. Various methods and tactics are reviewed to identify android malware. (Pallavi. K. et al.2020) employed the approach to identify malware in Android applications using a machine learning classifier that incorporated both static and dynamic features. They also extract the features using dynamic and static methodologies. Using the Environment for knowledge analysis to train machine learning classifiers and obtain the results using K-fold cross-validation.

A strategy was offered that included filter methods including Fisher Relief F-score, Detriment, and Maximum Relevance with the SVM envelope for the fewest data records needed to detect mobile malware (Dimitrics E, 2021). The method, which is based on mobile device user behaviour, improves cyber resilience. When malware is found on a mobile device, it is meant to automatically analyze, identify, and inform the user. A recently created Blockchain-based method for Android virus detection. The framework model of the Blockchain Consortium may identify the public chain shared by users and the consortium chain shared by test participants. The detection of harmful coding and the extraction of related data are its main concerns (Homayoun *et al.*, 2019). Outlined a procedure for classifying an unknown executable file once it has been analyzed.

## Static analysis

The strategy is put into practice using both static and dynamic analytical techniques. Static analysis is used to extract the binary code from the executable file and prepare the function set. As a result of the execution time analysis, the behaviour of executable files is examined. Each approach has its own significance, and each technique has certain benefits and drawbacks for identifying malware and other security holes. A technique of identifying malware utilizing static and dynamic properties retrieved from the executable file was proposed by (Awan, 2017). Based on static malware analysis, the authors presented their results. According to their approach, the authors employed import functions, grayscale image methods, and OPODODE NGR image techniques for the extraction procedure (Zhongzhi S., 2021). They come up with a powerful technique for spotting malware in Windows operating systems. Extraction of the executable file's many features. These characteristics are utilized as inputs for several machine-based classifiers that categorize malware executable files (Zhongzhi Shi, 2021).

(Arvind M., 2020) develops a malware detection framework by utilizing a certain collection of criteria that assist in determining whether an Android app belongs to the malware or benign class. The execution was carried out with the support of thirty different types of Android applications. They fill the gap by gathering applications that fall under thirty distinct categories, which are then utilized to reinforce and extend the findings and create a dataset. They collect 5,50,000 apk files, from Google’s play store,2 hacks, 3appchina, 4 Android, 5 mumayi,6 fan,7 slideme8, and the Panda app. 9 Among these 5,50,000 benign .apk files, 5,00,000 are distinct. Further, the features are extracted after deleting viruses-infected apps, as reported by Virus Total 10 and Microsoft Windows Defender.11

(Mohamed S.A, 2022) other studies investigate the performance of four machine learning classifiers that can find malware depending on dynamic and static features. (Long Wen et al. 2017) developed an algorithm to detect android malware called SVM, which is different from the traditional detection method. The proposed method shows a higher and lower error detection rate than the traditional approach. Dynamic tools like Scandroid, DroidRange, and vetdroid can identify malicious activities. Tools based on static analysis like Androgaurd, Pscout, FlowDroid, etc. Can detect and prevent the malicious application from being installed on the device, (Sangeeta R et al., 2019). Various data mining models are trained, and performance measures like accuracy and recall are assessed and compared (Mohamed S. ,2022).

## Dynamic analysis

Dynamic analysis approaches, as opposed than static analysis, govern the malware execution process. At runtime, collect, observe, and record malware behavior characteristics. The dynamic analysis approach is often executed in a secure virtual environment known as a sandbox. Cuckoo Sandbox (Cuckoo Sandbox, 2019) and CW Sandbox are two popular dynamic analysis sandboxes.

(CWSandbox, 2019) The primary purpose of the sandbox is to detect dangerous activity in malware and prevent it from harming the host system. Both static and dynamic analysis methodologies have benefits and drawbacks. The major benefit of static analysis over dynamic analysis is that it does not have the overhead cost of executing the program. Static analysis approaches, on the other hand, have limitations due to a lack of support for packaging and complicated obfuscated code. Dynamic analysis, as opposed to static analysis, can successfully evaluate bundled and disguised malware. This is because the virus must unpack itself while it is executing. As a result, the original and malicious code will be placed in the main memory. However, the fundamental disadvantage of dynamic analysis is that it takes time and resources. Malwaresamples must be evaluated independently, limiting the use of dynamic analysis in commercial applications.

## Machine learning techniques

For machine learning algorithms to be as successful as required (Fallah et al.,2019) benchmarking must occur first. The discovery of the specific family of harmful apps forms the basis of this method. The authors also demonstrate how combining techniques is essential to get consistent outcomes across platforms based on selected datasets or malware classifications. The authors suggest using network-based detection methods and machine learning in this circumstance. Machine learning should be applied both to supervised and unsupervised approaches to produce useful results that can be used in the decision-making process. However, However, the study does not specifically show how the machine-learning algorithms will deal with new malware types that have not yet been tested using available techniques. Therefore, algorithms must be continually updated to collect and detect the latest malware families and enable machine-learning techniques.

(Md. Shohel R., 2021) makes use of the DREBIN dataset to evaluate classifiers for Android malware detection that use tree-based machine learning. From the Drebin dataset, they chose 11,120 applications for testing, of which 5,560 had malicious files and the rest were benign. The accuracy of the SVM classifiers was 94%, whereas the accuracy of the random forest classifiers was superior to both at 97.24%. The model proposed by (Yeima et al. 2021) that uses static analysis based on Bayesian classification to provide indicators of potentially malicious activities yielded the most effective results, with TPR (True Positive Rate) of 90.6%, FNR (False Negative Rate) of 0.094%, an accuracy of 93.5%, and AUC (Area Under Curve) of 97.22%. Five supervised machine learning algorithms were used in the dynamic analysis model proposed by (Feizollah et al., 2019), with KNN generating the highest results with a TPR of 99.94% and a FPR of (False Positive Rate) of 0.06%.

# Chapter 3 - RESEARCH METHODOLOGY

This paper treats the Android application as an audio signal with unique properties that can be analyzed using Automated Signal Recognition techniques. Mel-frequency cepstral coefficients (MFCC), Bark Frequency cepstral coefficients (BFCC), and Normalized Gammachirp Cepstral Coefficients.

## Data collection

In this study, the dataset that was used in this study was collected from Drebin website and google play store.

DREBIN: Contains 5,560 malware files and 2201 benign APK files collected in the period of August 2010 to October 2012, and the Google Play store contains only benign. This is one of the most popular datasets for Android malware detection.

When evaluating the DREBIN dataset, we randomly sampled 5,600 clean files to match the number of malware samples in this dataset. Finally, the results of the machine learning algorithm tests on the two datasets are obtained and compared.

## Feature Engineering

In this process, we performed three audio feature extraction processes, which are Normalized Gammachirp Cepstral Coefficients (NGCC), Bark Frequency Cepstral Coefficients (BFCC), and Mel Frequency Cepstral Coefficients (MFCC).

* Drebin and Google Play Store datasets are collected and unzipped into different folders of these three audio signals.
* Both datasets have been cleansed to get rid of any duplicate and damaged APK files.
* To establish the distribution of the different samples in the dataset, data exploration and validation are performed.
* WAN files are created by converting APK files.
* In this paper, The Process of the Android application package (APK) to Waveform Audio File Format (WAV) and feature extraction was done using the method of (Tarwireyi P., 2022)

**Feature extraction**

APK files are used by Android to install various applications. APK files can be disassembled using the Android reverse engineering program Androguard. Two types of important files, AndroidManifest.xml and Classes.dex, are obtained by decompiling the APK file. The configuration data for Android applications, including the package name, permissions, and component information, is contained in an AndroidManifest.xml file (Jiyun Yang et al., 2022). In this paper we performed this algorithm to convert APK file to Audio.

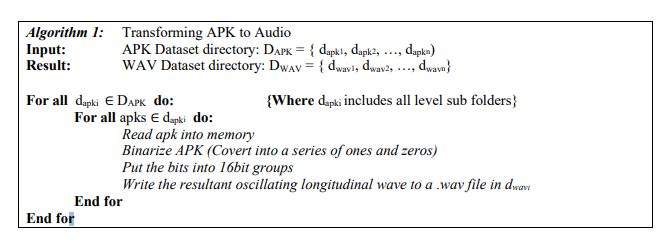


Figure 2: (P. Tarwireyi, 2022)

**Feature selection**

Different Android applications have different features that can be extracted from them (Jiyun Yang et al., 2022). To create a reliable malware detection system, not all features must be identified and examined. The features that are used by both benign software and malicious software have little to no impact on identifying benign software from malicious software (Jiyun Yang et al., 2022). As a result, they could be taken out of the feature set. Additionally, filter approaches have a cheap computational cost, can quickly remove redundant features, and are appropriate for large-scale feature sets. As a result, from the enormous original feature set, a filter method called feature selection based on the mean of weight is first used to extract the highly distinct features that can be useful in differentiating between benign and malware (Jiyun Yang et al., 2022). The weight of the features must be determined for the filter process. We design two distinct weight calculation methods—called frequency similarity weight and TF-IDF difference—to measure the importance of features according to the characteristics of different types of features, taking into account the various characteristics of various types of features in applications. Finally, a threshold is determined to filter out features of a certain type based on the mean weight of that type of feature.



Figure 3: Proposed methodology architecture diagram (Mercaldo. F & Santone. A, 2021)

### Normalized Gammachirp Cepstral Coefficients (NGCC)

The paper offers a feature extraction technique called Normalized Gammachirp Cepstral Coefficients (NGCC), which incorporates the characteristics of the Android application package, to increase resilience in noisy voice recognition.

Text

Description automatically generated

Equation 1: Calculate NGCC (Y. Zouhir & K. Ounis, 2017)

Equation 2: NGCC feature extraction ( Y.Zouhir & K. Ouni, 2016)

where M is the total number of NGCC coefficients, N is the number of auditory filterbank channels, and Log (Xk) is the energy output in logarithmic scale of the kth filter (k = 1, 2..., N). N and M are set to be 34 and 12, respectively, for the NGCC calculations.

Diagram

Description automatically generated

Audio signal

Figure 4: Flowchart of NGCC Feature Extraction (K. Ouni, 2016)

### Bark Frequency cepstral coefficient (BFCC)

Bark Frequency Cepstrum, which is based on the linear cosine transform of a log spectrum on a non-linear Bark scale of frequency, is a short-time power spectrum representation of a signal. Can be calculated using these equations.





Equation 3: Calculate BFCC (P. Tarwireyi, 2022)

Where f is the frequency obtained and bark seems to be the linear frequency of the waveform expressed in hertz, respectively.

This study uses the bark-frequency cepstral coefficients feature extraction technique highlighted below.

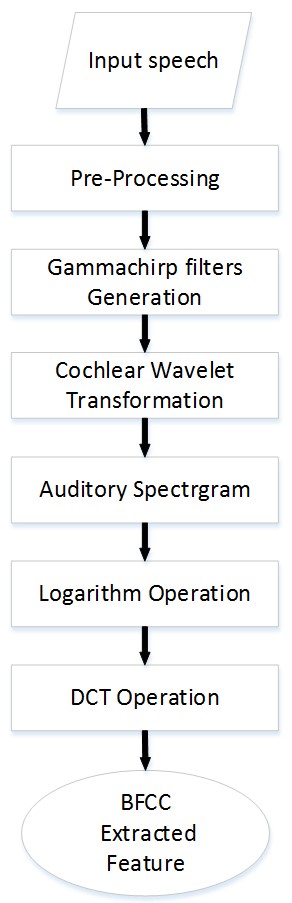


Figure 5: Flowchart BFCC Feature Extraction (K. Ouni, 2017)

### Mel Frequency Cepstral Coefficients (MFCC)

Windowing the signal, using the DFT, taking the log of the magnitude, warping the frequencies on a Mel scale, and then incorporating the inverse DCT are the main steps in the MFCC method of feature extraction. In this paper we use this audio feature extraction technique.

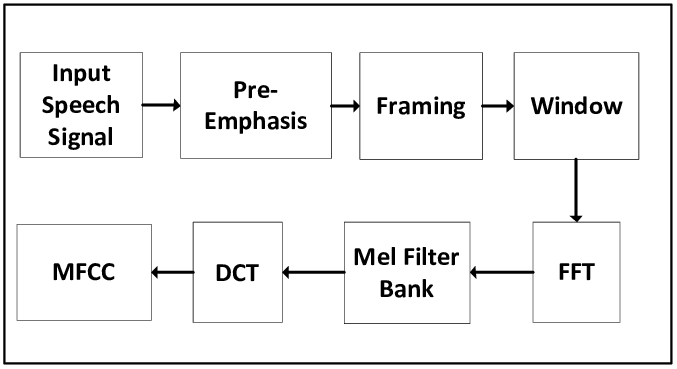


Figure 6: Flowchart of MFCC Feature Extraction (Magre S.B & Janse P.V, 2014)

## Machine learning algorithms evaluation

In this paper, nine algorithms are implemented. These Machine learning models are used to evaluate the performance of audio feature malware detection system.

* Logistic regression

LR shows the data and clarifies the link between one binary dependent variable and independent variables. It is performed on binary values based on a defined set of independent variables. It forecasts the probability of an event happening by fitting the data into a log function. To generate an output value p, input values x are linearly mixed with coefficient values b. As expected, the output numbers range between 0 and 1. data connected to the constant value coefficient b. As shown in, p is the outcome, b0 is an intercept term, and b1 is the coefficient of input value x.

**A picture containing chart

Description automatically generated**

* Decision Tree

A decision-support tool depicted by a tree-like graph or decision model that reflects the possible outcomes. It is a strategy in which conditional statements are used in an algorithm. DT is frequently employed in research activities or to solve categorization challenges. It aids in the identification of a strategy, especially for decision analysis, to achieve the goal more successfully. DT is a prominent classifier in machine learning. We divide the data into at least two homogeneous sets in DT. Then, using that data, perform decision analysis. This is done to create distinct groups based on independent variables or significant qualities. This classifier is commonly referred to as "decision trees," however in some platforms or stages, such as R, it is referred to by a different name called Classification and Regression Tree.

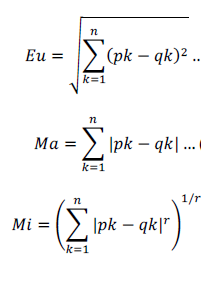
* Naïve Bayes

Bayesian theorem-based probabilistic classifiers with naive independence assumptions between predictors or features. The NB classifier assumes that the presence of one feature is unrelated to the presence of any other feature in the class. The NB model is simple to build and useful for big data sets, and the Bayes theorem provides a method for calculating posterior probability P(c|x) using probability (predictor probability) P(x|c), class prior probability P(c), and predictor prior probability P(c) (x).

****

* K-Nearest Neighbours (KNN)

Is employed in the solution of regression and classification issues. KNN is frequently used to solve classification difficulties. The KNN classifier saves all current instances before classifying new cases based on the majority votes of its neighbors. The case is allocated to the class that has the most in common with its k-nearest neighbors, as determined by the distance function. These distance functions are calculated using these equations as Euclidean Eu, Manhattan Ma, and Minkowski Mi.



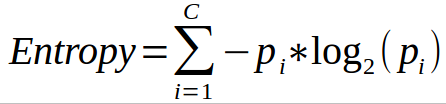
Whereas r represents the parameter, n represents the number of attributes or dimensions. pk and qk are the kth elements of the objects p and q, respectively.

* Support Vector Machine

Is a method for classification and regression. Every piece of data in SVM is displayed in n-dimensional space, with numerous dimensions equal to the number of features or attributes. Where n is the number of attributes. Each attribute's value is the value of a specific coordinate. After visualizing all the data items, classification was conducted by drawing a line or locating the closet hyperplane that distinctly separates two classes.

* Random Forest

Ensemble learning approaches can be used for regression, classification, as well as other applications. This works by creating many DT over the training period and displaying the mean prediction or mode of classes of the various trees. Random decision forests are sets of decision trees that overfit their training set. Every tree in the forest contributes to the classification. To categorize new cases based on relevant qualities. We determine the three "votes" for that class. As a result, the forest represents the classification of the case receiving the most votes. If the training set contains N items, a sample of N objects is drawn at random with replacement.



* Extra tree

Employs an ensemble supervised machine learning method that utilizes decision trees. Like the Random forests’ technique, the additional trees algorithm generates many decision trees, but the selection for each tree is unbiased and without replacement. This generates a dataset containing unique samples for each tree. For every tree, a specific number of features is chosen at random from the complete collection of features. The random selection of a split value for a feature is the most essential and distinguishing aspect of additional trees.

* Cat boost

Is a Supervised Machine Learning technique created by Yandex researchers and engineers for gradient boosting on decision trees.

* Bagging Decision Tree

This is accomplished by creating bootstrap samples from the training data set, then building trees on the bootstrap samples, collecting the output from all the trees, and predicting the outcome. Bagging entails creating many models using a subset of the data and then aggregating the predictions of the various models to reduce variation.

## Classification Metrics

Classification metrics assess model behavior or performance and show how accurate the classification is.

### Accuracy - is obtained by dividing the total number of predictions by the true positive predictions.

Text

Description automatically generated with medium confidence

*([Hatice K. Erdogan](https://medium.com/@kastanhatice?source=post_page-----688ca3a8743d--------------------------------), 2022)*

### Precision - Precision is defined as the ratio of true positives to total positives predicted.

Diagram

Description automatically generated with medium confidence*([Hatice K. Erdogan](https://medium.com/@kastanhatice?source=post_page-----688ca3a8743d--------------------------------), 2022)*

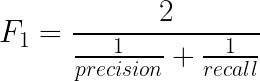
### Recall - is the ratio of true positives to all the positives in dataset.

Diagram

Description automatically generated

*([Hatice K. Erdogan](https://medium.com/@kastanhatice?source=post_page-----688ca3a8743d--------------------------------), 2022)*

### F1-Score - It combines accuracy and recall and is the harmonic mean of the two.



Equation 4: F1 score

*([Hatice K. Erdogan](https://medium.com/@kastanhatice?source=post_page-----688ca3a8743d--------------------------------), 2022)*

### Area under the curve - uses both false positive and true positive rates.



Equation 5: AUC

*([Hatice K. Erdogan](https://medium.com/@kastanhatice?source=post_page-----688ca3a8743d--------------------------------), 2022)*

# Chapter 4 - RESULTS AND DISCUSSION

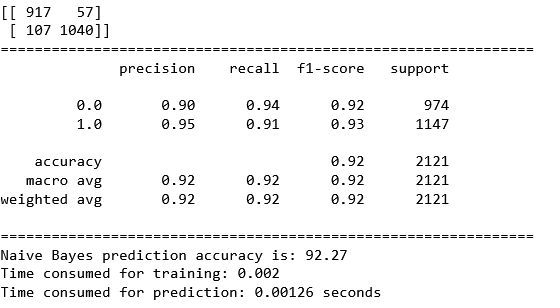
The results were implemented using Jupyter Notebook (Anaconda3) and the code was implemented in a machine learning pipeline and machine learning classifiers were implemented to evaluate performance. The machine learning classifiers were 9: Random Forest, Decision tree, Naïve Bayes, Support Vector Machine, Cat boot, Extra tree, Logistic Regression, and Bagging Decision tree.

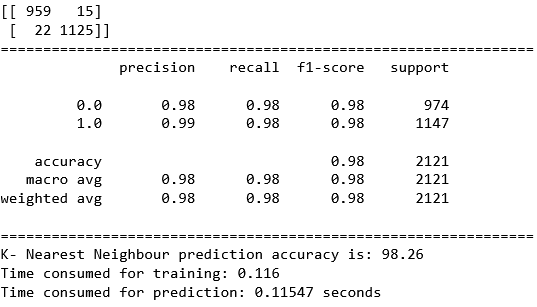
## Confusion Matrix

A table called a confusion matrix is used to describe how well a classification system performs. The effectiveness of the classification algorithms utilized in this study is represented and summarized in a confusion matrix.

### BFCC Confusion Metrics

LR BaggingDT KNN

 Table

Description automatically generated 

Decision Tree Random forest Naïve bayes

A screenshot of a computer

Description automatically generated with low confidence Table

Description automatically generated A screenshot of a computer

Description automatically generated with low confidence

SVM Extra Trees Cat boost

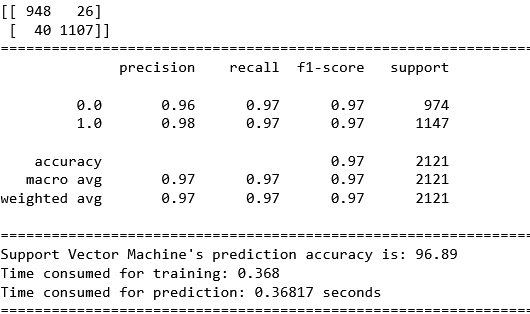
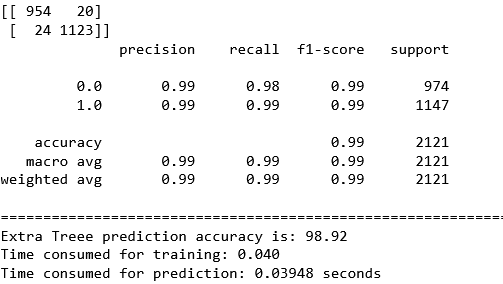
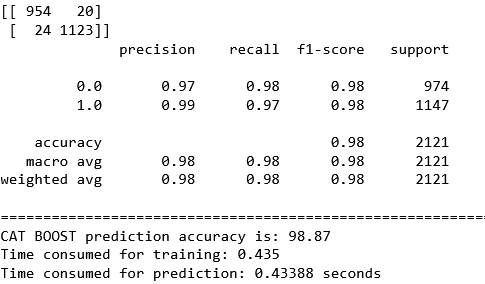
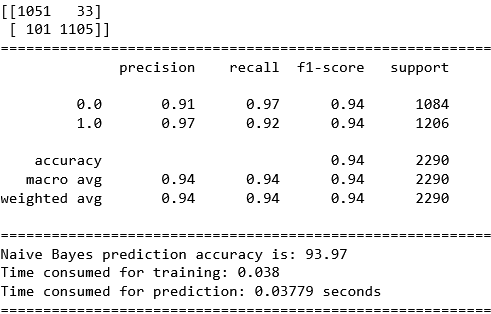
  

Figure 7: BFCC Confusion Metrics

### NGCC Confusion Metrics

LR BaggingDT Decision Tree

Table

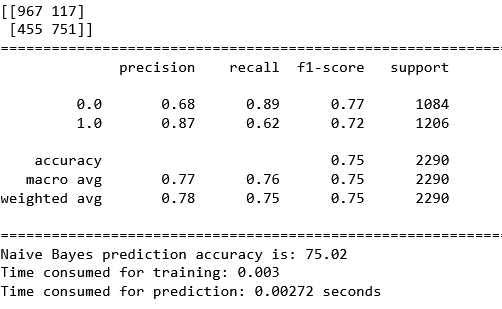
Description automatically generated  Table

Description automatically generated

Random Forest KNN Naïve Bayes

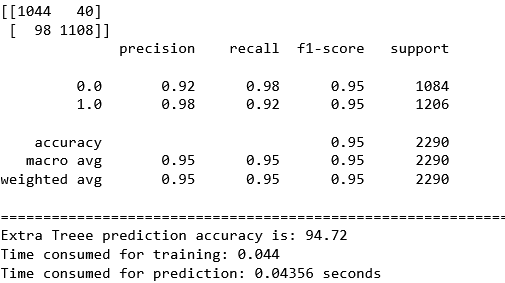
Table

Description automatically generated Table

Description automatically generated 

SVM Extra Trees Cat boost

A picture containing text, receipt, screenshot

Description automatically generated  Table

Description automatically generated

Figure 8: NGCC Confusion Metrics

### MFCC Confusion Metrics

LR BaggingDT Decision Tree

Table

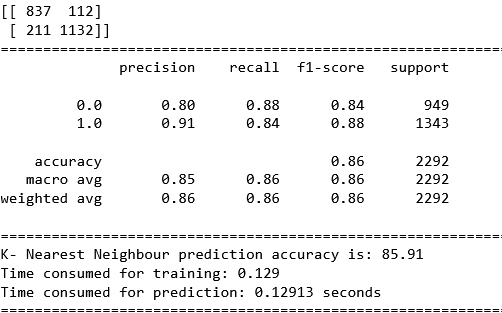
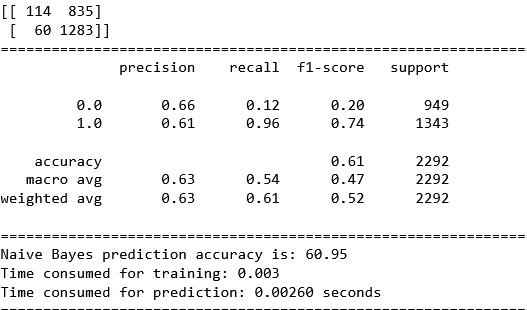
Description automatically generated Table

Description automatically generated Table

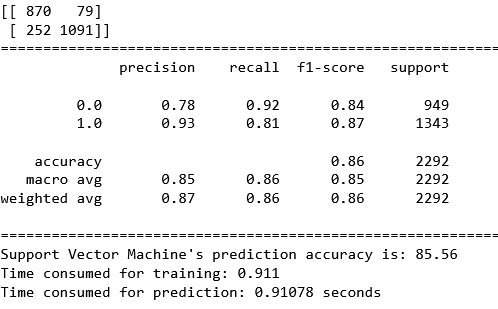
Description automatically generated

Random Forest KNN Naive Bayes

Table

Description automatically generated  

SVM Extra Trees Cat Boost

 Table

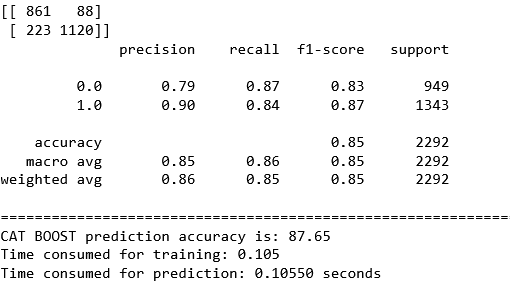
Description automatically generated 

Figure 9: MFCC Confusion Metrics

## Classification performance

Classification of accuracy, precision, recall, and area under the curve are determined to provide a comprehensive assessment of the classification accuracy of bark-frequency cepstral coefficients, Mel Frequency Cepstral Coefficients and Normalized Gammachirp Cepstral Coefficientsfor Android applications. Additionally, the training and test periods are calculated to evaluate how sophisticated the algorithms are. The following table displays the outcomes of tests on malware identification and classification using an 80-20 split on the Drebin dataset.

Table 2:BFCC models performance

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Classifier used | Test Accuracy | Precision | Recall | F1 Score | AUC | Train time  (s) | Test time  (s) |
| Extra Tree | 0.9901 | 0.99 | 0.99 | 0.99 | 0.99 | 0.040 | 0.03948 |
| SVC | 0.9882 | 0.97 | 0.97 | 0.97 | 0.99 | 0.034 | 0.348 |
| Naïve Bayes | 0.9872 | 0.785 | 0.90 | 0.78 | 0.98 | 0.002 | 0.00173 |
| Random Forest | 0.9872 | 0.98 | 0.98 | 0.98 | 0.96 | 0.029 | 0.02835 |
| KNN | 0.9825 | 0.97 | 0.98 | 0.97 | 0.98 | 0.116 | 0.11547 |
| Cat boost | 0.9768 | 0.98 | 0.98 | 0.98 | 0.97 | 0.435 | 0.43388 |
| Decision Tree | 0.9688 | 0.98 | 0.98 | 0.98 | 0.96 | 0.002 | 0.00154 |
| Logistic Regression | 0.9217 | 0.92 | 0.92 | 0.92 | 0.92 | 0.002 | 0.00126 |
| Bagging DT | 0.7713 | 0.98 | 0.98 | 0.98 | 0.97 | 0.036 | 0.03520 |

Table 3: NGCC models performance

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Classifier used | Test Accuracy | Precision | Recall | F1 Score | AUC | Train time  (s) | Test time  (s) |
| KNN | 0.9471 | 0.94 | 0.94 | 0.94 | 0.95 | 0.152 | 0.15244 |
| SVC | 0.9458 | 0.94 | 0.94 | 0.94 | 0.74 | 0.456 | 0.45619 |
| Naïve Bayes | 0.9441 | 0.75 | 0.73 | 0.72 | 0.95 | 0.003 | 0.00272 |
| Random Forest | 0.9423 | 0.94 | 0.95 | 0.95 | 0.94 | 0.35 | 0.03462 |
| Extra Tree | 0.9379 | 0.95 | 0.95 | 0.95 | 0.94 | 0.044 | 0.04356 |
| Cat boost | 0.9296 | 0.95 | 0.95 | 0.94 | 0.93 | 0.055 | 0.05473 |
| Decision Tree | 0.9021 | 0.94 | 0.95 | 0.95 | 0.90 | 0.002 | 0.00212 |
| Logistic Regression | 0.8982 | 090 | 0.90 | 0.90 | 0.90 | 0.002 | 0.00168 |
| Bagging DT | 0.7502 | 0.95 | 0.95 | 0.95 | 0.76 | 0.038 | 0.03779 |

Table 4: MFCC Models performance

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Classifier used | Test Accuracy | Precision | Recall | F1 Score | AUC | Train time  (s) | Test time  (s) |
| Naïve Bayes | 0.8747 | 0.61 | 0.54 | 0.47 | 0.88 | 0.003 | 0.00260 |
| Random Forest | 0.8730 | 0.87 | 0.88 | 0.87 | 0.88 | 0.036 | 0.03646 |
| KNN | 0.8704 | 0.85 | 0.86 | 0.86 | 0.88 | 0.129 | 0.12913 |
| SVC | 0.8590 | 0.85 | 0.86 | 0.85 | 0.86 | 0.911 | 0.91078 |
| Extra Tree | 0.8555 | 0.87 | 0.89 | 0.88 | 0.86 | 0.048 | 0.04803 |
| Cat boost | 0.8516 | 0.87 | 0.88 | 0.87 | 0.86 | 0.105 | 0.10550 |
| Logistic Regression | 0.8228 | 0.82 | 0.82 | 0.82 | 0.82 | 0.002 | 0.00163 |
| Decision Tree | 0.8180 | 0.82 | 0.82 | 0.82 | 0.81 | 0.002 | 0.00221 |
| Bagging DT | 0.6095 | 0.82 | 0.83 | 0.82 | 0.54 | 0.038 | 0.03818 |

## Benchmarking with other works.

The performance statistics of the various models considered in this study are displayed in Tables 2 to 4. The below table shows the results comparison of this study and the work of other authors that used the Drebin dataset to detect Android malware.

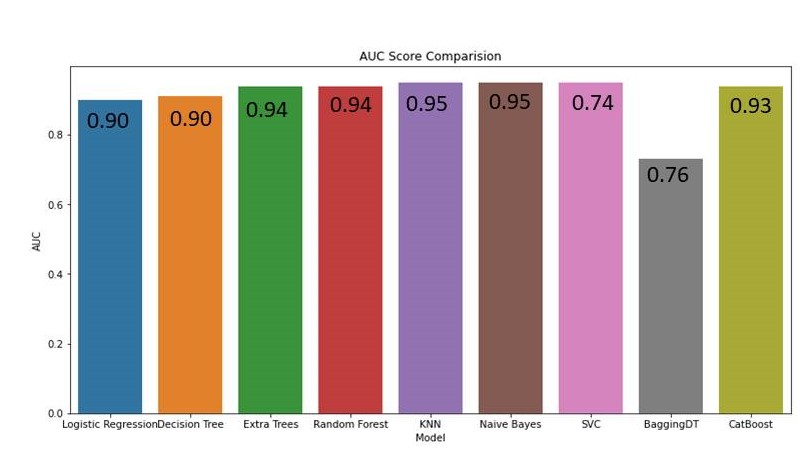
Table 5: Comparison of the study and other authors

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Authors | Models | Accuracy | Precision | Recall | F1-Score |
| Rahul Y, 2022 | Random Forest | 0.981 | 0.943 | 0.860 | 0.900 |
| Qing Wu, 2020 | K-Nearest Neighbour | 0.976 | 0.969 | 0.874 | 0.918 |
| CHENGLIN Li, et al. 2022 | Naïve Bayes | 0.977 | 0.962 | 0.992 | 0.977 |
| Mohamed S.A, 2022 | Random Forest | 0.973 | 0.973 | 0.785 | 0.869 |
| Md Shohel R, 2018 | Random Forest | 0.938 | 0.950 | 0.930 | 0.950 |
| This study  (BFCC) | Extra Tree | 0.990 | 0.99 | 0.99 | 0.990 |
| This study (NGCC) | KNN | 0.947 | 0.94 | 0.94 | 0.94 |
| This study (MFCC) | Naïve Bayes | 0.874 | 0.61 | 0.54 | 0.47 |

* This study compares BFCC, MFCC, and NGCC metrics results with the other authors’ work that uses the same dataset, with the purpose of achieving the most effective audio feature that can be used in future as audio feature technique to detect malware on android operating system. In the comparison, BFCC has a high accuracy score of 0.99, a precision of 0.99, a F1-Score of 0.99 and a recall of 0.99
* NGCC and MFCC have most effective results, but (Rahul Y., 2022) has better results than them.
* In other words, the BFCC feature is most likely to detect malicious applications on Android.

## Comparison graphs

Below are the graphs of the Area under the curve and the accuracy for different feature extraction techniques. The area under the curve shows how much the model is capable of distinguishing between classes. It also shows the accuracy is used to identify which model is most effective at detecting Android malware.



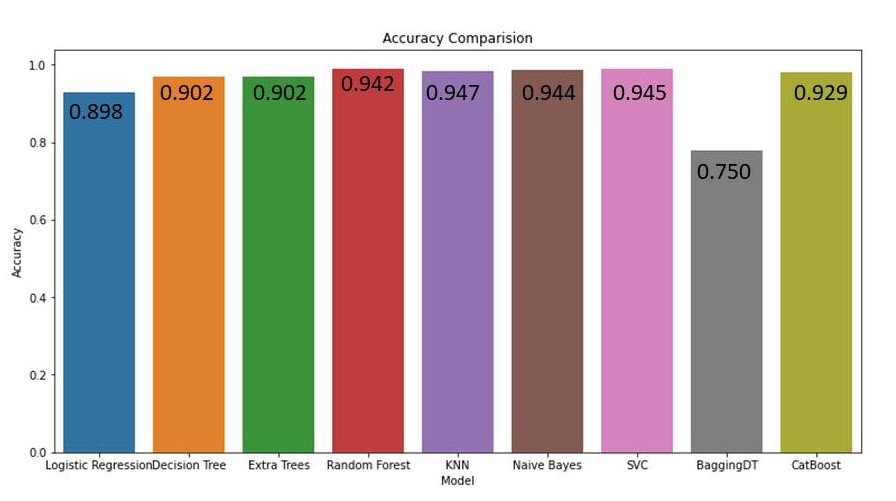
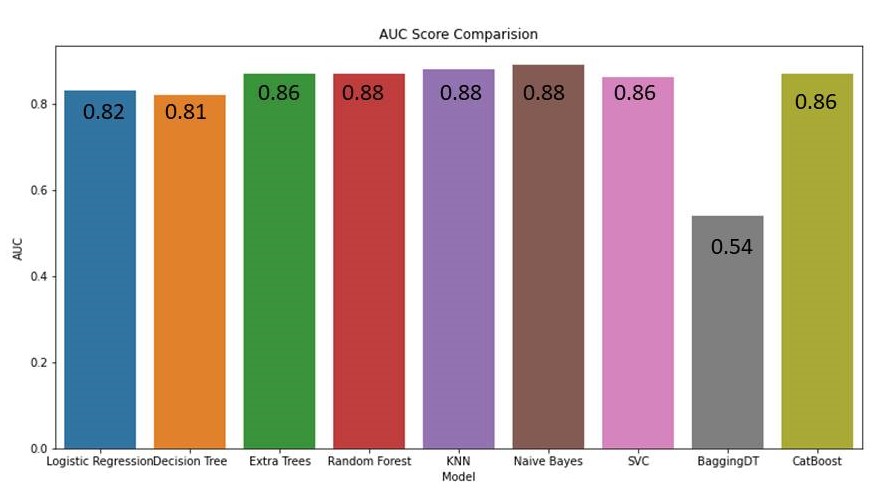


Figure 10: NGCC graphs for Accuracy and AUC



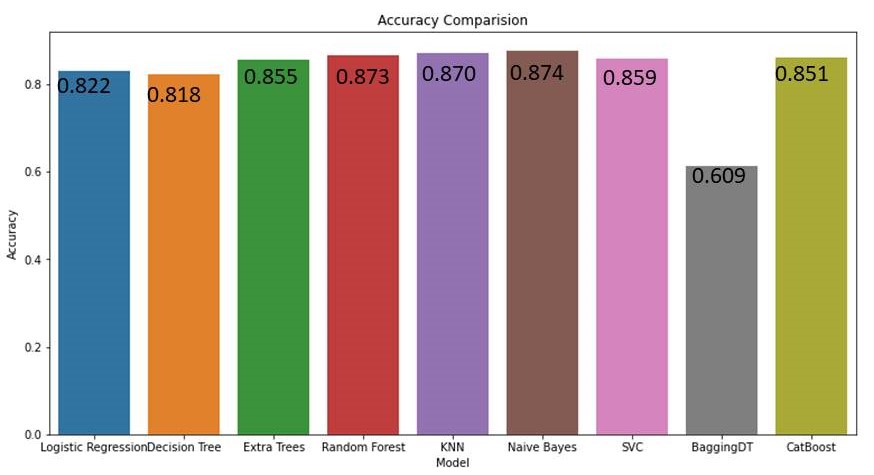


Figure 11: MFCC graphs for Accuracy and AUC

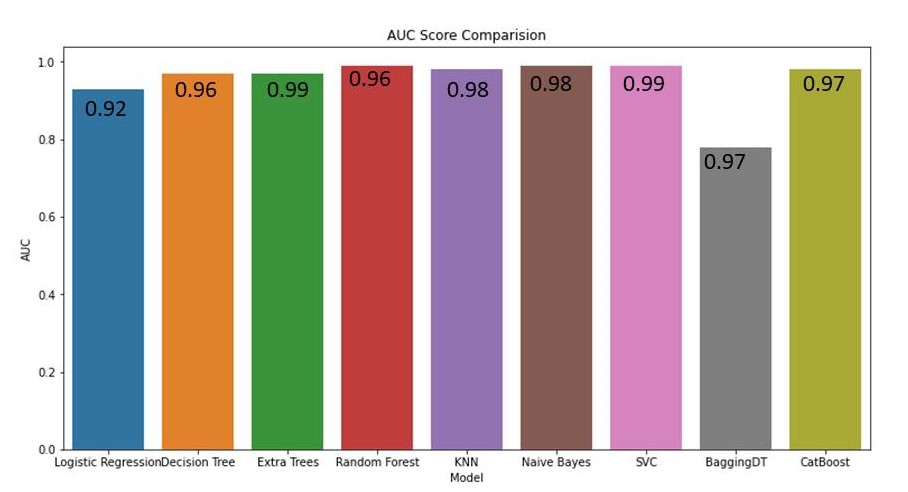
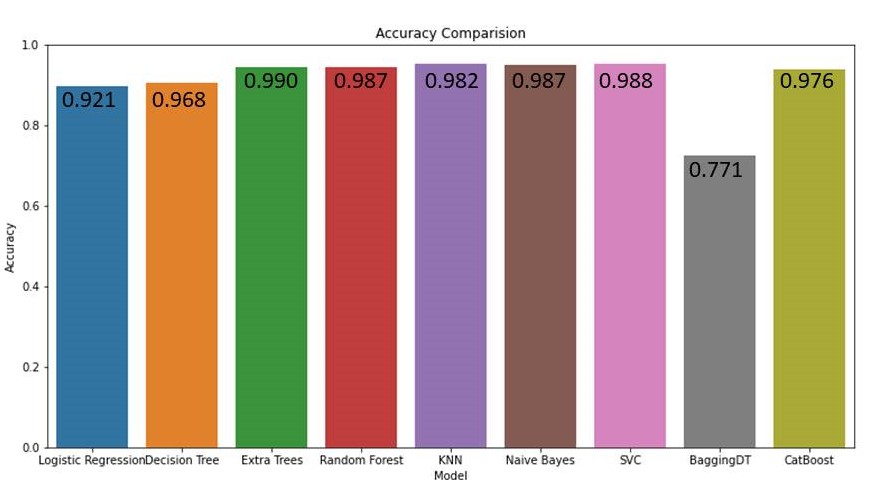
 

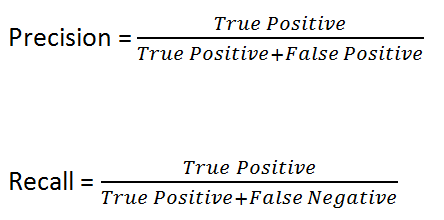
Figure 12: BFCC graphs for Accuracy and AUC

The audio features graphs for accuracy and area under the curve are shown. On BFCC the algorithm that has the highest accuracy is Extra Tree with 99.0%, with the area under the curve of 99.0%. The NGCC feature obtained the accuracy of 95.0% with the area under the curve of 95.0%. The last feature is MFCC with the accuracy of 86.0% and area under the curve of 88.0%. Meaning BFCC feature is the one that can detect malware on android application than other features.

## Precision-Recall Graphs

Since accuracy alone is insufficient to comprehend the performance of classification models, precision and recall metrics are employed to evaluate machine learning algorithms.

Precision-recall can be calculated using the following formulas.



Equation 6: Precision- recall

*(C. Riggio, 2019)*

Graphical user interface

Description automatically generated with low confidence

Shape

Description automatically generated with medium confidence

Figure 13: BFCC precision-recall

Figure 14: BFCC precision-recall

Figure 15: MFCC Precision-recall

A picture containing shape

Description automatically generated

Figure 16: NGCC precision-recall

The BFCC feature with extra trees algorithm is probably the better option. This is due to its precision score of 99.0%, indicating that it has the most accurate ratio of malicious apps that are classified correctly. Audio features have algorithms that provide high precision and recall rates, which means they return a high number of accurately labelled results.

# Chapter 5 – Conclusion

In this paper, nine machine-learning algorithms were used to evaluate the performance of the proposed method for Android malware detection. As the above results indicate, the most accurate classifier for the prediction of android malware is Extra Trees for Bark Frequency Cepstral Coefficients which achieved the highest score of 99.0%. After the comparison of the existing literature results with this study, we find that the Bark Frequency Cepstral Coefficients feature extraction technique is the most accurate among other Authors, which means that an audio signal can be used for android malware detection.

The contribution of this paper to Android malware detection is that we treat Drebin datasets as audio signal features. which is the first thing in this dataset to use such a format. For the future of android malware, More research is required to determine which feature is most accurate in audio-based features.

# REFERENCES

[1] Ashawa, M. and Morris, S. (2019a) ‘Analysis of Android Malware Detection Techniques: A Systematic Review’, *International Journal of Cyber-Security and Digital Forensics*, 8(3), pp. 177–187. Available at: https://doi.org/10.17781/p002605.

[2] Kuan, T.-W. *et al.* (2016) ‘A robust BFCC feature extraction for ASR system’, *Artificial Intelligence Research*, 5(2). Available at: https://doi.org/10.5430/air.v5n2p14.

[3] Li, C. *et al.* (2019) ‘Android Malware Detection Based on Factorization Machine’, *IEEE Access*, 7, pp. 184008–184019. Available at: https://doi.org/10.1109/access.2019.2958927.

[4] Lindorfer, M., Neugschwandtner, M. and Platzer, C. (2015) ‘MARVIN: Efficient and Comprehensive Mobile App Classification through Static and Dynamic Analysis’, *2015 IEEE 39th Annual Computer Software and Applications Conference* [Preprint]. Available at: https://doi.org/10.1109/compsac.2015.103.

[5] Mahindru, A. and Sangal, A.L. (2020) ‘MLDroid—framework for Android malware detection using machine learning techniques’, *Neural Computing and Applications*, 33(10), pp. 5183–5240. Available at: https://doi.org/10.1007/s00521-020-05309-4.

[6] Mahindru, A. and Singh, P. (2017) ‘Dynamic Permissions based Android Malware Detection using Machine Learning Techniques’, *Proceedings of the 10th Innovations in Software Engineering Conference* [Preprint]. Available at: https://doi.org/10.1145/3021460.3021485.

[7] Mishra, J. *et al.* (2021) ‘Duplicates in the Drebin Dataset and Reduction in the Accuracy of the Malware Detection Models’, *2021 26th IEEE Asia-Pacific Conference on Communications (APCC)* [Preprint]. Available at: https://doi.org/10.1109/apcc49754.2021.9609892.

[8] Odusami, M. *et al.* (2018) ‘Android Malware Detection: A Survey’, *Communications in Computer and Information Science*, pp. 255–266. Available at: https://doi.org/10.1007/978-3-030-01535-0\_19.

[9] Pomianos, G. *et al.* (2003a) ‘Recent advances in the automatic recognition of audiovisual speech’, *Proceedings of the IEEE*, 91(9), pp. 1306–1326. Available at: https://doi.org/10.1109/jproc.2003.817150.

[10] Rana, Md.S., Rahman, S.S.M.M. and Sung, A.H. (2018) ‘Evaluation of Tree Based Machine Learning Classifiers for Android Malware Detection’, *Computational Collective Intelligence*, pp. 377–385. Available at: https://doi.org/10.1007/978-3-319-98446-9\_35.

[11] Sulapas Mapa, J. (2018) ‘Android Untrusted Detection with Permission Based Scoring Analysis’, *International Journal of Information Technology Convergence and Services*, 8(3/4), pp. 01–12. Available at: https://doi.org/10.5121/ijitcs.2018.8401.

[12] Ul  Hassan, C.A., Khan, M.S. and Shah, M.A. (2018c) ‘Comparison of Machine Learning Algorithms in Data classification’, *2018 24th International Conference on Automation and Computing (ICAC)* [Preprint]. Available at: https://doi.org/10.23919/iconac.2018.8748995.

[13] Wen, L. and Yu, H. (2017) ‘An Android malware detection system based on machine learning’, *AIP Conference Proceedings* [Preprint]. Available at: https://doi.org/10.1063/1.4992953.

[15] Wu, Q. *et al.* (2020) ‘MVIIDroid: A Multiple View Information Integration Approach for Android Malware Detection and Family Identification’, *IEEE MultiMedia*, 27(4), pp. 48–57. Available at: https://doi.org/10.1109/mmul.2020.3022702.

[16] Youchao Dong (2017) ‘Android Malware Prediction by Permission Analysis and Data Mining’, *University of Michigan - Deep Blue* [Preprint]. Available at: http://deepblue.lib.umich.edu/bitstream/2027.42/136197/1/YouchaoDong\_Thesis\_0327.pdf.

[17] Yuan, B. *et al.* (2020) ‘Byte-level malware classification based on markov images and deep learning’, *Computers &Amp; Security*, 92, p. 101740. Available at: https://doi.org/10.1016/j.cose.2020.101740.

[18] Yumlembam, R. *et al.* (2022) ‘IoT-based Android Malware Detection Using Graph Neural Network With Adversarial Defense’, *IEEE Internet of Things Journal*, pp. 1–1. Available at: https://doi.org/10.1109/jiot.2022.3188583.

[19] Zouhir, Y. and Ouni, K. (2016) ‘Feature Extraction Method for Improving Speech Recognition in Noisy Environments’, *Journal of Computer Science*, 12(2), pp. 56–61. Available at: https://doi.org/10.3844/jcssp.2016.56.61.

[20] Zwane, S., Tarwireyi, P. and Adigun, M. (2019) ‘Ensemble Learning Approach for Flow-based Intrusion Detection System’, *2019 IEEE AFRICON* [Preprint]. Available at: https://doi.org/10.1109/africon46755.2019.9133979.