National College of Ireland

Android Malware Detection using Machine Learning and Convolutional Neural Network

MSc Research Project
Cyber Security

Sai Hari Parthiban
Student ID: 22169148

School of Computing
National College of Ireland

Supervisor: Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sai Hari Parthiban		
Student ID:	22169148		
Programme:	MSc in Cyber Security	**Year:**	2023
Module:	MSc Research Project		
Supervisor:	Jawad Salahuddin		
Submission Due Date:	05/04/2024		
Project Title:	Android Malware Detection using Machine Learning and Convolutional Neural Network		
Word Count:	6661	**Page Count:**	25

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

Signature: Sai Hari Parthiban

Date: 04/04/2024

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 Malware Detection using Machine Learning and Convolutional Neural Network Algorithm

Sai Hari Parthiban
22169148

Abstract

The fast evolution of Android mobile devices has opened new possibilities for convenience and usefulness, but it has also given rise to Android malware, which poses a major danger to user security such as data theft. To discriminate between benign and malicious apps, effective detection systems are essential. Machine learning and Deep learning have emerged as a potent connect in this domain, capable of swiftly assessing Android application packages (APK files) and properly classifying applications. In this study, we examine the efficacy of various machine learning models and a deep learning model, with the Random Forest, Extra Tree, XGBoost, Stacking Classifier and Convolutional Neural Network (CNN), in detecting Android malware also Scikit Framework has been used for detecting malware in this research. Between these models, CNN stands out as the best performer, with respect to its precision, memory, and F1 scores to determine their correctness and dependability. This superior result demonstrates its capacity to reliably detect and categorise Android apps, emphasizing its importance in improving mobile security. This report offers critical insights into the state of Android malware detection using machine learning, offering a path forward to enhance Android security and shield users from potential security threats in the ever-evolving landscape of mobile technology.

Keywords: Android malware, Machine learning, CNN, APK files, Scikit Framework

1 Introduction

As of fast growth of mobile intelligent terminals, Android has developed the most widely used computing platform on smartphones according to Meijin et al (2022). The widespread utilization of Android-based mobile devices has heralded a new era of unprecedented ease and connectedness, revolutionizing the way people live, work, and communicate. Android malware, such as Trojans, viruses, ransomware, and spyware, poses a serious danger, attaining detection and mitigation an urgent and vital problem. With its ability to process and analyze enormous datasets, machine learning has emerged as a powerful tool in the field of Android virus detection. Machine learning and deep learning models may categorize Android application packages (APK files) as benign or malicious by managing features derived from

these packages. The accurate detection of harmful applications is critical for improving mobile security and protecting users from possible dangers. Numerous research methodologies have been presented to address the problems posed by Android malware. This research evaluates machine learning models and a deep learning model in the context of Android malware detection. It aims on classifiers such as Random Forest, Extra Tree, XGBoost, Stacking Classifier and CNN Model, all of which have indicated promise in discriminating between safe and dangerous applications. This paper sheds light on the current status of Android malware detection and lays the groundwork for future breakthroughs in mobile security and the privacy and data protection of Android users. The graph of android malware rise is shown as below[1] :

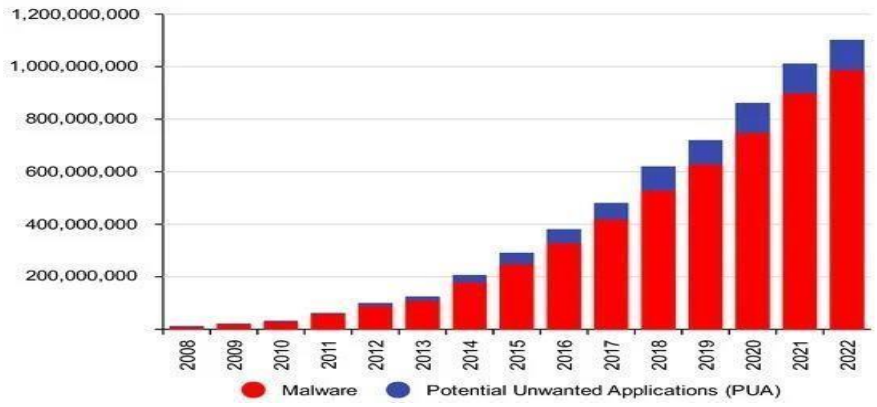

Figure 1 Malware rise.

1.1 Research Questions

The research questions for this report are as follows:

1. Which machine learning models, including Random Forest, Extra Tree, XGBoost, and Stacking Classifier demonstrate the highest accuracy and reliability in distinguishing between benign and malicious Android applications?
2. Which model is more efficient when compared between machine learning and the deep learning model used in the android malware detection?

The goal of this research is to design and test machine learning models and a deep learning model for Android malware detection to develop security and safeguard users from potential risks. The main purpose of this approach is to identify malware in Android applications also

[1] https://www.mdpi.com/2073-8994/15/3/677

the major goal is to reliably categorize Android applications as benign or malicious using a wide variety of classifiers such as CNN, Random Forest, Extra Tree, XGBoost, and a Stacking Classifier. The study intends to present an effective way of discriminating between safe and possibly hazardous apps by obtaining high accuracy, recall, and F1 scores, therefore combining to the overall security of Android users. The ultimate aim is to determine the effective model for Android malware detection and to provide insights into future developments and research paths in mobile security.

2 Related Work

2.1 Machine Learning in Different Security Domains

According to Shaukat et al (2020) emphasise the importance of machine learning in improving cyber security measures and combating hackers' shifting techniques. While Vanjire and Lakshmi (2021) concentrate on mobile security and use machine learning to discover malware vulnerabilities, which aligns with the wider issue of advance digital security. Butt et al (2020) investigate cloud computing security, proposing the use of machine learning techniques to lower vulnerabilities and safeguard data. Cui et al (2018) highlight the revolutionary potential of machine learning in IoT applications, to improve security, network management, and data analytics. Kotenko et al (2018) presents a methodology for enhancing security monitoring in the perspective of the mobile Internet of Things by combining Big Data processing with machine learning. Regardless of their specific domains of application, these evaluations highlight the critical role of machine learning in providing efficient and proactive solutions to address security, data analytics, and intelligent decision-making, consequently contributing to the total goal of improving digital security and performance.

Table 2.1: Comparative Analysis in Technology and Security Domains

Study	**Domain**	**Proposed Approach**	**Challenges Addressed**	**Key Results/Contributions**
Shaukat et al., 2020	Cyber Security	Machine learning for threat detection and response	Evasion of conventional security systems	Enhanced security measures for cyber threats

Vanjire and Lakshmi, 2021	Mobile Security	Machine learning for malware vulnerability detection	Mobile security and malware infections	Improved security for Android mobile devices
Butt et al., 2020	Cloud Computing Security	Machine learning algorithms for data protection	Security vulnerabilities in cloud computing	Enhanced cloud security using ML
Cui et al., 2018	Internet of Things (IoT)	Machine learning for IoT applications and analytics	Security, network management, data analytics	Intelligent IoT applications and improved security
Kotenko et al., 2018	Big Data Processing and Security	Big Data processing and machine learning for security	Security monitoring in mobile IoT	Improved security monitoring and Big Data analytics

2.2 Models for Android Malware Detection

2.2.1 Random Forest Classifier Model

Rana et al (2018) investigated the use of machine learning classifiers for Android malware detection, achieving an accuracy rate of more than 94 percent. Similarly, Agrawal and Trivedi (2021) recognised the increasing sophistication of Android malware and executed machine learning classifiers to develop detection accuracy, with Random Forest establishing to be the most accurate classifier. Koli, (2018) proposed "RanDroid," a machine learning-based malware detection solution with a remarkable accuracy of 97.7 percent, stressing the insertion of various characteristics from Android apps into their model. Furthermore, Jung et al (2018) presented a fresh strategy by concentrating on the selection of relevant Android APIs, with their Random Forest classifier achieving an astounding accuracy of 99.98 percent when employing a refined set of APIs. Finally, Zhu et al (2018) established the exponential rise in Android malware and presented an ensemble machine learning system for Android malware detection, reaching an accuracy rate of 89.91 percent by applying a random forest classifier. These studies, examined together, highlight the necessity of machine learning in Android malware detection, as well as the usefulness of other factors, such as API usage and permissions, in obtaining high detection accuracy.

2.2.2 XGBoost Classifier Model

Giannakas et al (2022) conducted a careful investigation of machine learning models, improving and examining 27 machine learning algorithms alongside a Deep Neural Network (DNN) to determine the best accurate model. Chen et al (2018) adopted a additional visual method, transforming Android malware's dex files into photos and classifying them using the XGBoost algorithm. They obtained an astounding 99.14 percent accuracy. Darus et al (2019) concentrated on producing grayscale pictures from Android APK files using the GIST descriptor and XGBoost for classification, demonstrating the higher performance of their technique. Palsa et al (2022) used XGBoost and very randomised trees algorithms to classify Android malware with great accuracy and sensitivity. Finally, Ling et al (2019) used Ant Colony Optimization (ACO) to tune XGBoost settings, improving detection accuracy while reducing false positives.

2.2.3 Stacking Classifier Model

Multiple studies have identified the necessity for strong detection systems that not only classify these threats properly but also categorize the precise type of attack for effective prevention. To address this urgent issue, researchers have offered several novel techniques. Wang et al (2022) developed the MFDroid framework to detect Android malware via stacking ensemble learning. They learned an outstanding F1-score of 96.0 per cent by using seven characteristic selection methods and logistic regression as a meta-classifier. Xie et al (2023) observed the issue of dynamic obfuscation techniques used by Android malware. They presented the GA-StackingMD system, which used the Genetic Algorithm and Stacking to advance detection accuracy and attained outstanding results on two datasets. Furthermore, Shafin et al (2021) introduced a two-layer Machine Learning detection model based on Ensemble Learning and Stacked Generalization, reaching 99.49 percent accuracy in detecting Android smartphone attack types, outperforming previous research on the same dataset.

2.2.4 Convolutional Neural Network Model

Yadav et al (2022) employed stacked recurrent neural networks (RNNs) and convolutional neural networks (CNNs) to recognize and classify Android malware using image-based versions. Lee et al (2019) concentrated on the covering of Android applications as well. Their method used stacked RNNs and CNNs to discover correlations between obscured text patterns in package names and certificate owner names, developing the obfuscation resistance of their feature extraction system.

Study	Domain	Proposed Approach	Challenges Addressed	Key Results/Contributions
Rana et al., 2018	Android Malware Detection	Machine learning classifiers for Android malware detection	Addressing Android malware threats, detection accuracy	Achieved an accuracy rate exceeding 94%, demonstrating the effectiveness of machine learning classifiers. Random Forest classifier used.
Agrawal and Trivedi, 2021	Android Malware Detection	Machine learning-based malware detection methodology	Overcoming malware detection limitations, accuracy	Identified Random Forest as the most accurate classifier.
Koli, 2018	Android Malware Detection	RanDroid: Machine learning-based malware detection system	Comprehensive feature extraction, high classification accuracy	Achieved an accuracy of 97.7%, a significant contribution to Android malware detection.
Jung et al., 2018	Android Malware Detection	Machine learning methodology focusing on Android APIs	Improved API-based detection, high accuracy	Achieved an accuracy of 99.98%, emphasizing the relevance of specific APIs. Random Forest classifier used.
Zhu et al., 2018	Android Malware Detection	Ensemble machine learning system for Android malware detection	Android malware proliferation, cost-effective alternatives	Achieved an accuracy of 89.91%, providing a cost-effective approach for detection. Random Forest classifier used.
Giannakas et al. (2022)	Android Malware Detection	Experimented with 27 machine learning algorithms and a DNN, optimized with Optuna.	Android malware detection, model selection.	Identified DNN with four layers as the best model, achieving 86% prediction accuracy. Feature analysis using SHAP.
Chen et al. (2018)	Android Malware Detection	Converted Android malware's dex files into images and used XGBoost for classification.	Efficient Android malware detection.	Achieved a remarkable 99.14% accuracy, demonstrating the effectiveness of their visual approach.

Darus et al. (2019)	Android Malware Detection	Utilized grayscale images from Android APK files, GIST descriptor, and XGBoost for classification.	Enhanced Android malware classification.	Demonstrated the efficiency of their approach, especially when compared to other algorithms.
Palsa et al. (2022)	Android Malware Detection	Employed XGBoost and extremely randomized trees for malware detection.	Improved malware detection efficiency.	Achieved high accuracy and sensitivity, with successful integration into the MLMD program for automation.
Ling et al. (2019)	Android Malware Detection	Introduced Ant Colony Optimization (ACO) to optimize XGBoost parameters for malware detection.	Parameter optimization in Android malware detection.	Improved detection accuracy while reducing false positives, enhancing cybersecurity measures for Android devices.
Wang et al. (2022)	Android Malware Detection	Stacking Ensemble Learning with Feature Selection	Detection accuracy, traditional feature limits	F1-score of 96.0% and improved detection
Yadav et al. (2022)	Android Malware Detection	Deep Learning with RNNs and CNNs for Image Analysis	Handling obfuscation, improving accuracy	Outperformed existing models and classifiers
Xie et al. (2023)	Android Malware Detection	GA-StackingMD with Genetic Algorithm and Stacking	Dynamic obfuscation, hyperparameter optimization	Strong performance on two datasets
Lee et al. (2019)	Android Malware Detection	Stacked RNNs and CNNs for String Pattern Correlations	Addressing obfuscation challenges	Robust feature extraction and obfuscation resilience
Shafin et al. (2021)	Android Malware Detection	Ensemble Learning with Stacked Generalization	Accurate prediction and classification	Exceptional accuracy in classifying attack types

2.3 Scikit Framework

The Scikit-learn framework was used in this finding to present a solid foundation for assessing and classifying Android applications, distinguishing between benign and malicious apps. Scikit-learn includes a comprehensive set of machine learning approaches for evaluating and

evaluating models such as Random Forest, Extra Tree, XGBoost, plus Stacking Classifier. The flexibility and convenience of the use of Scikit-learn permitted the trial and judgment of many models, which substantially contributed to understanding the efficiency of machine learning in Android malware detection. Furthermore, its user-friendly interface advanced in the discovery of dataset constraints, underlining the need for additional diverse data to improve model resilience. As the research delves into future avenues of investigation, such as deep learning techniques, feature optimization, and real-time threat intelligence integration, Scikit-versatility lays the groundwork for further advancements in enhancing Android security and safeguarding users in the volatile world of mobile technology.

3 Research Methodology

3.1 Methodology

The workflow of the proposed methodology involves the following stages:

1. **Imported Libraries:** Shutil for file operations, os and zip file for file management, pandas for data manipulation, seaborn and numpy for data visualisation and numerical processing, scikit-learn for machine learning, pickle for object serialisation, and Plotly for interactive data visualisation were all imported. In addition, the stat module was loaded for file attribute manipulation. These collections collaborate to make data collection, preparation, analysis, and visualisation easier, as well as machine learning model development and report graphical user interface (GUI) production possible.
2. **Data Collection:** The information was gathered by doubling the dataset and then unzipping the resulting files. The dataset was then translated into a binary classification format, with 0 representing benign target class values and 1 demonstrating malicious target class values. This was a critical stage in practicing the data for analysis and machine learning. It supported the study to recognize between benign and malicious data, setting the framework for subsequent feature extraction as well as model training.
3. **Feature Extraction:** For feature extraction, the Androguard library was utilised, with a aim on obtaining application permission features from APK files. Androguard class 'androguard.core.bytecodes.apk.APK' was used to explain the APKs. The app's permissions were of particular relevance, and they were collected using the Androguard library's 'get permissions ()' method. These extracted permission attributes were critical in describing the behaviour of Android apps and operated as the foundation for later categorization. A total of 388 features were retrieved, which were then processed and

translated into a numerical representation for use in machine learning algorithms for classification tasks.

4. **Data Preprocessing:** The data preparation process includes many critical phases that were employed throughout the dataset generation. Initially, images of each sign were captured, and background subtraction techniques were used to remove the signs from their backgrounds, ensuring that only critical information was kept. For preprocessing the data this study has used androguard for first generating features using this and then converting the categorical value to the numerical values.
5. **Data Splitting (Training and Testing the Model):** The dataset was divided into two subsets during the Data Splitting phase: a training set and a testing set, with an 80 per cent to 20% ratio. The training set was used to train the machine learning models to learn patterns from data, while the investigating set was used to evaluate the models' performance and generalisation capabilities. We guaranteed that the models were not overfitting to the training data and could make reasonable predictions on unknown data by separating the data in this way.
6. **Model Training:** Numerous machine learning systems were used during the Model Training phase, with Random Forest Classifier, Extra Tree Classifier, XGBoost Classifier, and a Stacking Classifier built of an ensemble of XGBoost and Random Forest, with an Extra Trees Classifier acting as the meta-classifier. To realize patterns from the retrieved characteristics and app permissions, several algorithms were applied to the training dataset. The models improved their ability to discriminate between benign and malicious Android applications through recurrent training. This training step paved the way for the future categorization of unknown applications in the testing phase, allowing the models to make correct predictions and accurately identify possible security issues.
7. **Model Evaluation:** The Model Evaluation step examined the performance and efficacy of the taught machine learning models. The accuracy score offered an overall assessment of how successfully the models classified Android apps. The classification report included specific metrics for each class (benign and malicious) such as accuracy, recall, F1-score, and support, providing insights into the models' performance on actual classes. The confusion matrix graphically represented true positives, true negatives, false positives, and false negatives, assisting in the evaluation of model behaviour and implying possible areas for development.

3.2 Data Visualization

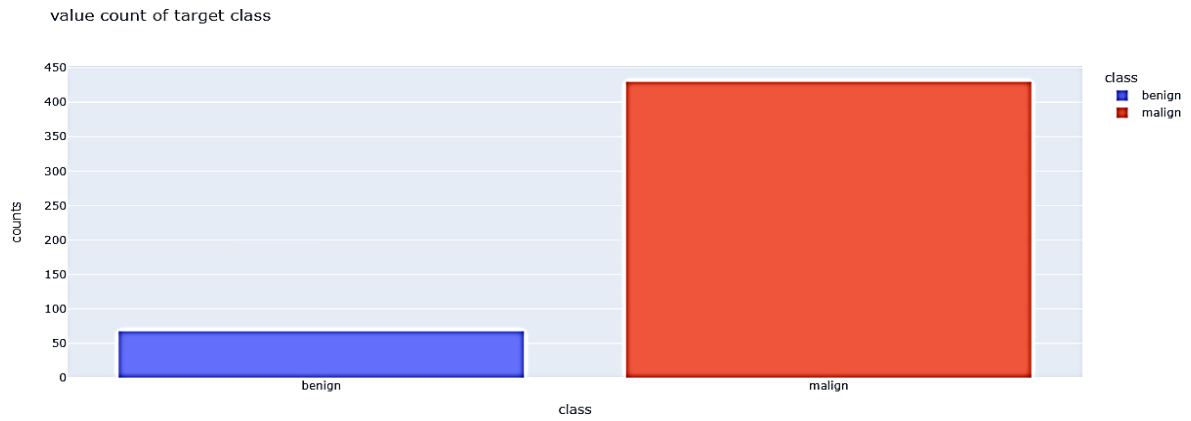

Figure 3.1: Bar Graph - Class Distribution of Benign and Malign Android Applications

Figure 3.1 is a bar graph or count plot that depicts the distribution of the target classes in a dataset where the two classes are imbalanced. The target classes in this example are tagged "benign" and "malign," which are often associated with Android applications (APK files). The classes are represented on the graph's x-axis, with "benign" and "malign" being the two categories under examination. The y-axis, on the extra hand, displays the number of instances in each class, ranges from 0 to 450. The goal of this graph is to prove how many examples or samples exist in each class. In a setting with unbalanced classes, you would generally anticipate the majority class (in this example, presumably "benign") to have a substantially greater count than the minority class ("malign"). This graph may help you understand the class distribution and analyze the level of class imbalance, which can have an impact on model training and performance.

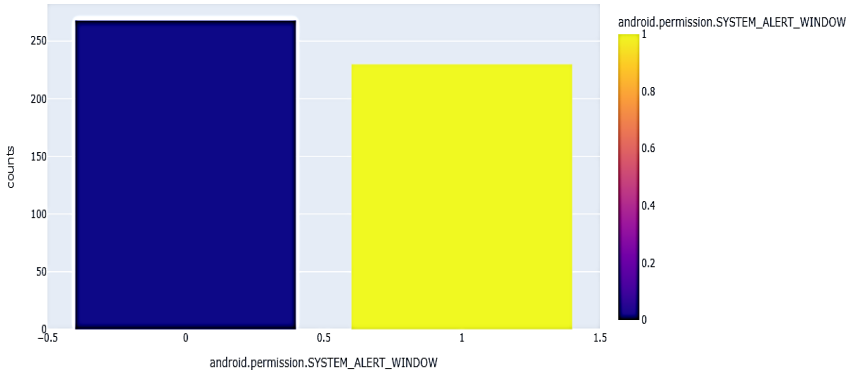

Figure 3.2: Bar Graph

The distribution of the Android permission "android.permission.SYSTEM ALERT WINDOW," which is normally binary in form, is seen in Figure 3.2. (0 for absence, 1 for

presence). This permission is important for understanding Android application behavior, specifically their ability to display overlay windows on the device's screen, and its distribution is depicted in the bar graph.

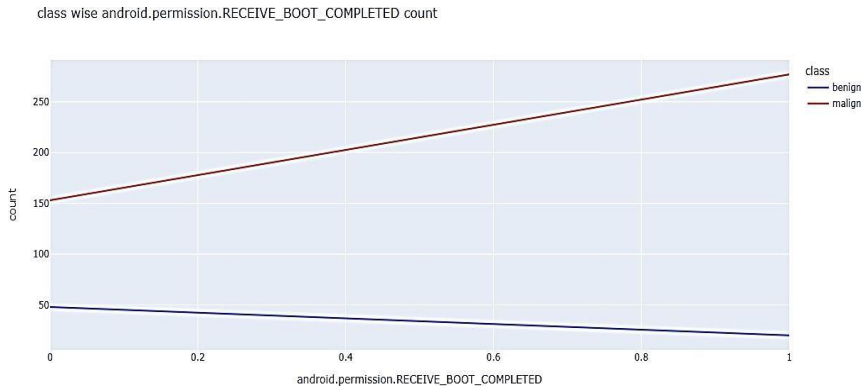

Figure 3.3: Comparison of Benign and Malign Android Applications

Figure 3.3 is a line graph that shows a visual comparison of two classes in a dataset, "benign" and "malign." The graph employs various colours to differentiate between the classes, with "benign" representing blue and "malign" representing red. The graph's x-axis usually shows a range of numbers, which can be counts, features, or any other important variable. It varies from 0 to 300 in this example. The major goal of this line graph is to show how the "benign" and "malign" classes vary or connect to the variable along the x-axis clearly and comparably.

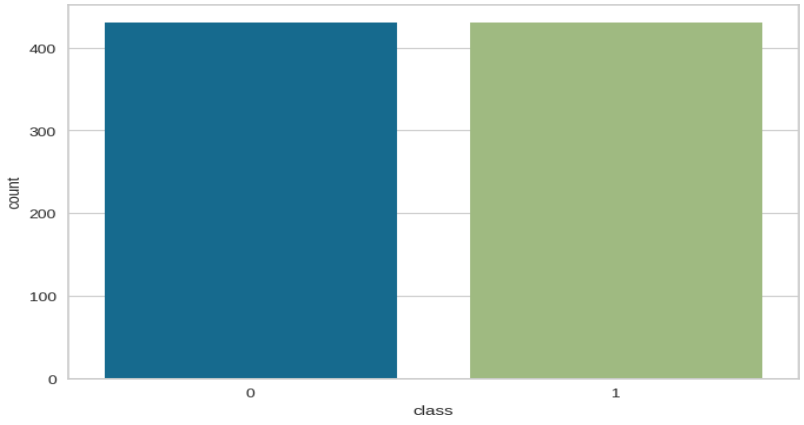

Figure 3.4: SMOTE Balancing Effect

Figure 3.4 depicts the use of the Synthetic Minority Over-sampling Technique (SMOTE) for data balancing, with an emphasis on class distribution in a count plot. The plot's x-axis displays the classes, with "0" being the dominant class and "1" representing the minority class. The counts of instances or samples are displayed on the y-axis, which ranges from 0 to 400. SMOTE

is a strategy for dealing with class imbalance in datasets, notably in machine learning. It creates a more balanced distribution by producing synthetic samples for the minority class.

3.3 Dataset Description

Here, “CIC-InvestndMal2019” dataset from the University of New Brunswick, involves of gathered samples of Android applications (APK files) classified as benign or malicious (or "malign"). The malign class exhibits four sub-categories to produce a binary classification problem: Adware, Ransomware, Scareware, SMS Malware, and PremiumSMS. This dataset's major focus is on app permissions, with a particular emphasis on analyzing the permissions sought by Android applications. These permissions are crucial for understanding the apps' behavior and any security concerns. The dataset may be used for a variety of machine learning and security activities, such as recognizing Android applications as benign or dangerous, extracting features from APKs, and investigating the links between permissions and app behavior. 0 is for benign and 1 is for malign.

3.4 List of Models

List of Models given below:

1. **Random Forest Classifier:** A powerful collaborative realizing approach that combines several decision trees to improve classification accuracy. It has been found to be successful in malware identification and it is good for dealing with a sort of features.
2. **Extra Tree Classifier:** The Extra Tree Classifier, like Random Forest, is an ensemble technique that grants additional variability in its base models by randomising the decision tree construction process. This can result in advanced generalisation and resistance for overfitting.
3. **XGBoost Classifier:** XGBoost is a gradient boosting system well-recognised for its remarkable implementation and efficiency when dealing with enormous datasets. It is often used in malware detection due to its ability to detect complex patterns and irregularities in data.
4. **Stacking Classifier:** The Stacking Classifier is a meta-model that combines predictions from many base models to get a concluding result. By combining the capabilities of many algorithms, it can increase classification accuracy in the perspective of Android malware detection. To construct a robust ensemble, it combines XGBoost and Random Forest as foundation models, with Extra Trees as the meta-classifier.
5. **Convolutional Neural Network (CNN):** Emerging a Convolutional Neural Network (CNN) model for Android malware detection includes input layers for app features,

convolutional layers for feature extraction, pooling layers for dimensionality reduction, and completely connected layers for sorting. For training, use a labelled dataset of both malicious and benign applications. Use strategies like as dropout to improve generalisation. For effective parameter updates, use an Adam optimizer and an appropriate loss function, such as cross-entropy, to optimise the model. After the model has been trained, put it to use on Android smartphones by incorporating it with security apps to find and stop any attacks. This will strengthen the foundation for mobile cybersecurity.

3.5 Model Selection and Performance Evaluation: A Comparative Approach

In the study of Smmarwar et al (2022), utilized the CIC-InvesAndMal2019 dataset, employing decision tree, random forest, and support vector machine (SVM) models for Android malware detection, achieving accuracies of 91.80%, 91.32%, and 82.33%, correspondingly. Building upon this, our study employs the same dataset, exploring a diverse set of machines learning models and one Deep learning model. Remarkably, our CNN model surpasses previous accuracies, achieving an impressive 95.34%. We performed and achieved better accuracy than prior work.

4 Design Specification

The Design Specification serves as a comprehensive manual detailing the essential aspects, requirements, and specifications of the Android malware detection system project. It outlines the software, hardware, and user interface components essential for the project's design and implementation. Software specifications include the selection and integration of libraries and tools for data collection, preprocessing, feature extraction, model training, and evaluation, such as Androguard, Pandas, Scikit-learn, and Plotly. Various machine learning methods and a deep learning, including Random Forest, Extra Tree, XGBoost, Stacking Classifier, and CNN, along with their respective components, are detailed. The data-gathering procedure involves copying, unzipping, and transforming the dataset into a binary classification issue, focusing on APK file permissions and extracting 388 features. Techniques like SMOTE oversampling and label encoding are employed to handle null values, extraneous columns, and class imbalance. Hardware requirements emphasize sufficient CPU capabilities for efficient dataset processing and machine learning tasks. Additionally, GUI creation using the Tkinter toolkit is highlighted, enabling functionalities

such as APK file uploading, classification result display, feature information presentation, and model performance metric visualization.

5 Implementation

The Implementation provides a comprehensive overview of the Android malware detection system's construction and execution, following the methodology outlined in the preceding chapter. The dataset is achieved, pre-processed, and investigated using specialized software libraries and tools such as Androguard, pandas, scikit-learn, and Plotly. Data collection involves downloading and unzipping the dataset from a specified URL and converting it into a binary classification format to distinguish between benign and malicious applications. Feature extraction involves extracting 388 features from APK files, with a focus on app permission features. Data preparation tasks include handling null values, removing extraneous columns, and addressing class imbalance using the Synthetic Minority Over-sampling Technique (SMOTE). Categorical variables are encoded numerically for efficient machine learning model functioning. Hardware infrastructure requirements are specified to ensure proper management of dataset processing and model operations. The Tkinter library is utilized to develop a user-friendly graphical user interface (GUI), allowing users to upload APK files for classification, view results, access feature information, and analyze model performance metrics. This chapter effectively demonstrates the transition from project design to real-world implementation, facilitating the creation of an Android malware detection system incorporating powerful machine learning algorithms and an intuitive interface.

5.1 GUI (Tkinter) Output Screens

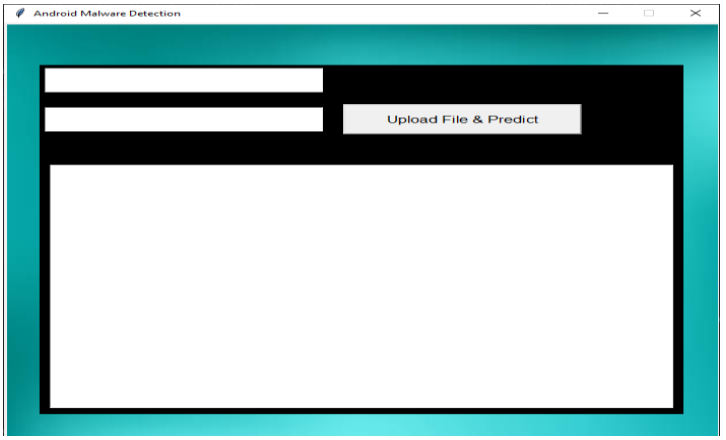

Figure 5.1: Android Malware Detection Web Application Interface

In Figure 5.1, the Android malware detection web application interface is displayed, featuring a user-friendly design. Notably, it includes an "Upload File" button, allowing users to submit

Android application packages (APK files), and a "Predict" option. This streamlined interface enhances user interaction and facilitates the seamless prediction of malware within uploaded files.

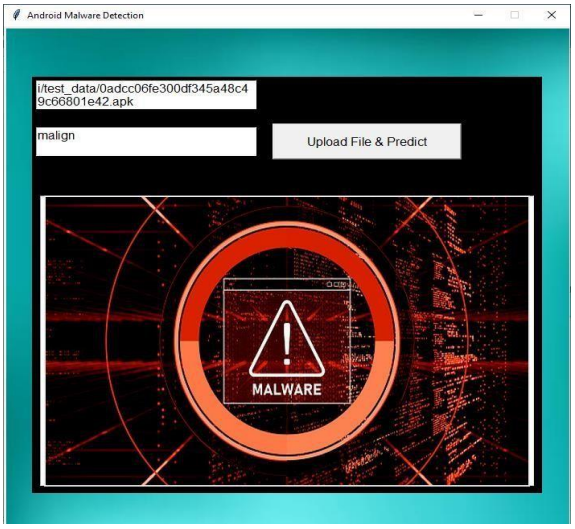

Figure 5.2: Malign Detection Result Display

In Figure 5.2, the web application decisively identifies the uploaded file as malicious, displaying a prominent "malware" label beneath. Accompanying this classification is a visual representation, featuring an illustrative malware image, effectively conveying the detected threat.

6 Evaluation

6.1 Random Forest Classifier Model

The Random Forest Classifier is a general and often used algorithm recognized for constructing accurate and resilient results in Raczko and Zagajewski (2017). It works by assembling a set of decision trees, each of which distinctly analyses distinctive features retrieved from Android applications (APK files). The Random Forest Classifier uses diversity and mitigates overfitting by aggregating predictions from numerous decision trees, boosting generalization to latest, unseen data. This algorithm excels in detecting subtle patterns and abnormalities in the dataset, making it particularly useful in Android malware detection, where distinguishing between benign and malicious apps is based on the study of different factors such as permissions, behaviors, and characteristics.

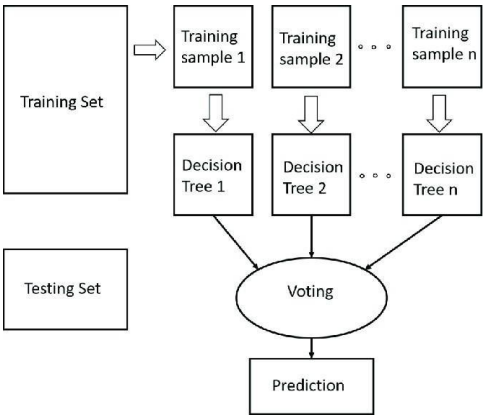

Figure 6.1.1: Random Forest Classifier Architecture by Le et al (2020)

The Random Forest Classifier's accuracy score of 81.97 per cent suggests that this machine learning model is qualified of accurately categorizing Android products as benign or malicious with an accuracy rate of nearly 88 per cent. This suggests that the model can successfully classify the characteristics of these applications in the situation of Android malware detection, distinguishing between those that are safe (benign) and those that may cause security problems (malicious).

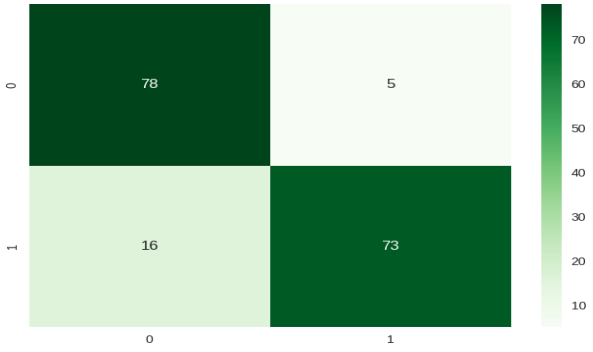

Figure 6.1.2: Confusion Matrix

6.2 Extra Tree Classifier Model

The Extra Tree Classifier is a powerful algorithm recognized for producing reliable and accurate results. The Extra Tree Classifier, like the Random Forest, is an ensemble approach that constructs numerous decision trees to assess information taken from Android applications (APK files). What distinguishes it, is the addition of randomization to the tree-creation process. Because the decision trees it produces are purposely designed more varied, the classifier is less prone to overfitting because of the increased unpredictability in feature selection and node splitting as described in Chen et al (2011).

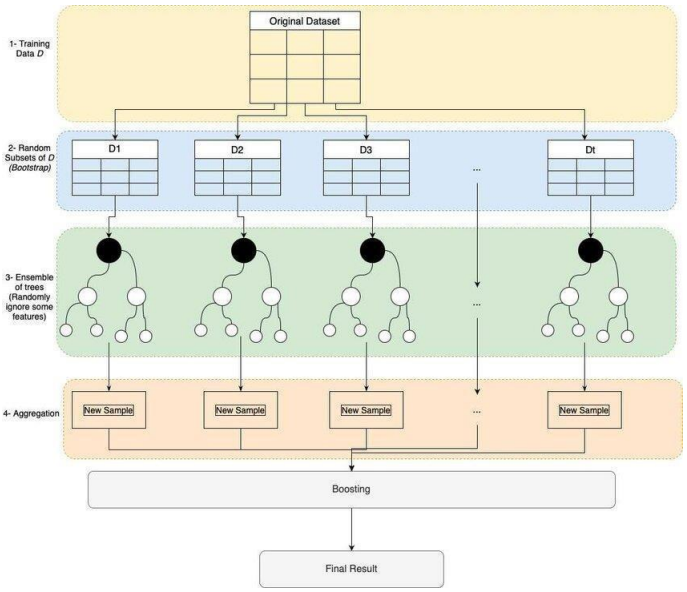

Figure 6.2.1: Extra Tree Classifier Architecture by Zaher et al (2023)

The Extra Tree Classifier's accuracy score of 89.53 per cent reveals that this machine-learning model is capable of reliably categorizing Android applications as benign or malicious with a high degree of precision, reaching an accuracy rate of nearly 89 per cent. This refers to the model's ability to discriminate between safe (benign) and possibly hazardous (malicious) applications in the context of Android malware detection. A greater accuracy score implies a better ability to forecast correctly.

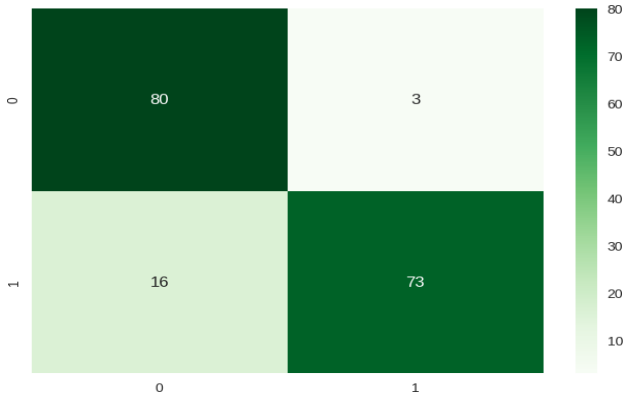

Figure 6.2.2: Confusion Matrix

6.3 XGBoost Classifier Model

The XGBoost (Extreme Gradient Boosting) classifier is regarded as a significant and strong algorithm. It is particularly valued for its remarkable performance in processing complex datasets and its ability to detect subtle patterns and abnormalities in the context of Android applications (APK files). XGBoost employs a gradient-boosting framework, in which an

ensemble of decision trees is iteratively produced, with each tree meant to correct the faults of the prior ones as mentioned in Trizoglou et al (2021). This iterative technique allows for the progressive increase of forecast accuracy as well as the capture of detailed linkages within the data.

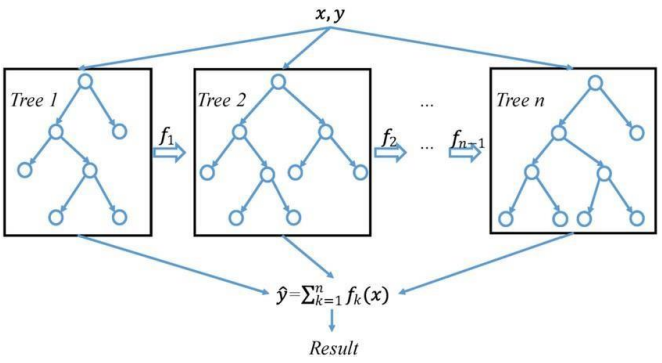

Figure 6.3.1: XGBoost Classifier Architecture by Wang et al (2019)

The XGBoost Classifier's accuracy score of 92.02 percent suggests that this machine learning model excels at properly categorizing Android applications as benign or malicious, with an accuracy rate of nearly 92 percent. This means that the model is particularly effective at discriminating between safe (benign) and possibly hazardous (malicious) applications in the context of Android malware detection. A high accuracy score indicates that the XGBoost Classifier is an excellent choice for this task, exceeding the previously described Random Forest and Extra Tree Classifiers.

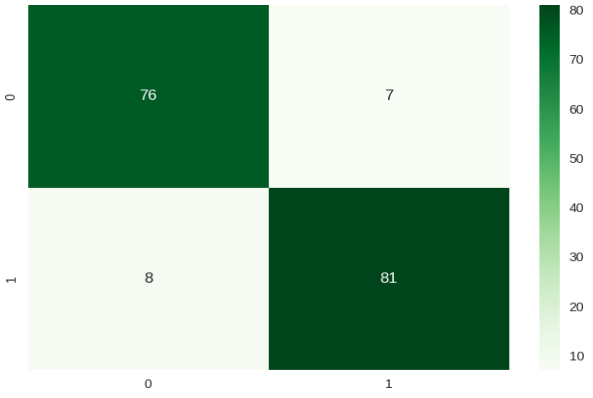

Figure 6.3.2: Confusion Matrix

6.4 Stacking Classifier Model

The Stacking Classifier, combines two prevailing machine learning models, the XGBoost (Extreme Gradient Boosting) Classifier and the Random Forest Classifier, with an Extra Trees Classifier portion as the meta-classifier, representing a sophisticated ensemble approach. This stacking methodology is meant to leverage the strengths of multiple models and improve

classification accuracy. The individual classifiers, XGBoost and Random Forest, each excel in obtaining complex patterns and subtleties within the Android application dataset, especially when believing features like permissions and behaviors. By combining these two models, the Stacking Classifier effectively harnesses their complementary capabilities.

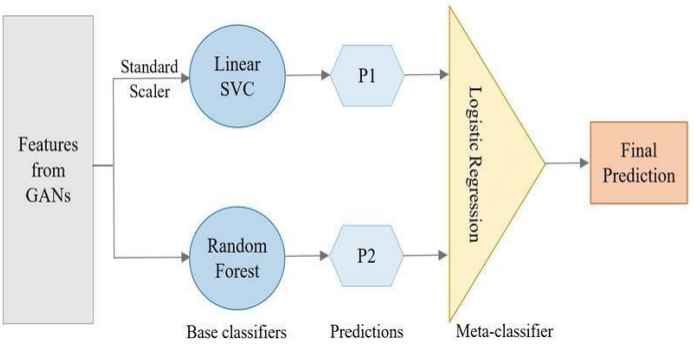

Figure 6.4.1: Stacking Classifier Architecture by Yadav et al (2021)

The Stacking Classifier, combines the XGBoost Classifier and Random Forest Classifier as basis models and leverages the Extra Trees Classifier as the meta-classifier, has an oddly high accuracy score of 93.76 per cent in categorizing Android apps as benign or malignant. This classifier is the combination of other 2-3 models but still, it does not give the highest accuracy at all.

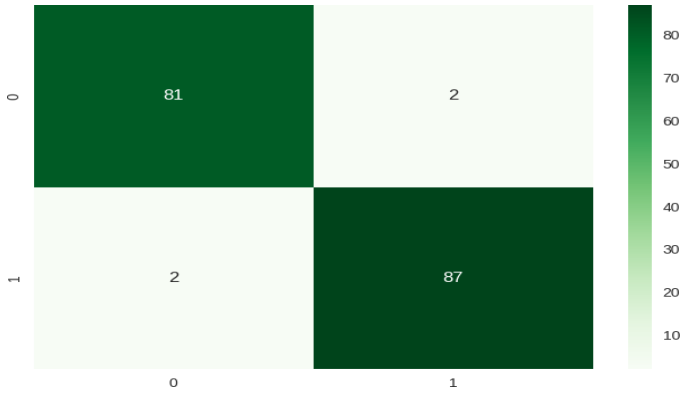

Figure 6.4.2: Confusion Matrix

6.5 CNN Model

Convolutional layers capture spatial patterns after input layers analyze app characteristics in a Convolutional Neural Network (CNN) architecture for Android malware detection. Pooling layers improve computing performance by lowering dimensionality. After that, fully linked layers analyze high-level information for categorization, separating safe apps from dangerous ones. ReLU activation functions can be used to add non-linearity. Use strategies such as

dropout to improve model generalization. Utilizing an Adam optimizer and a cross-entropy loss function, train the CNN on a labelled dataset. After the model has been trained, install it on Android devices to enable accurate and efficient malware detection by analyzing the features of the apps. This is one of the best model having highest accuracy upto 95 per cent.

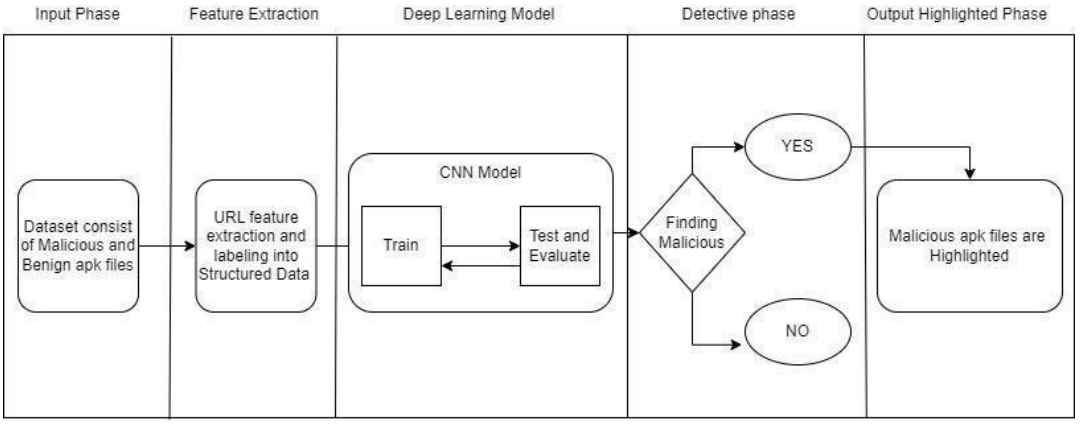

Figure 6.5.1: CNN Architecture

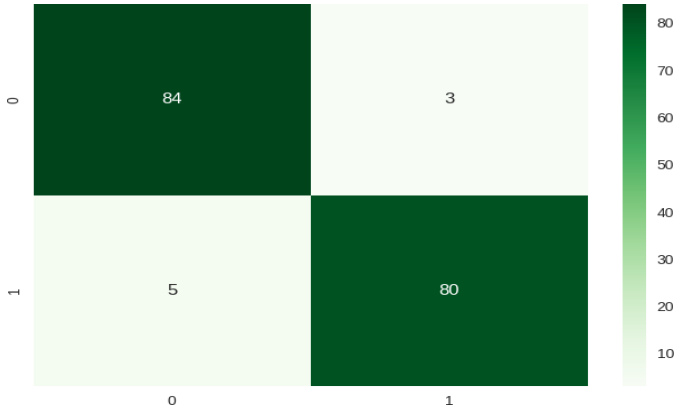

Figure 6.5.2: Confusion Matrix

Figure 6.5.3 depicts the training and validation accuracy curves for a Convolutional Neural Network (CNN). The blue line represents the training accuracy, showcasing the model's performance on the training data over epochs. The red line represents the validation accuracy, indicating how well the model generalizes to unseen data. A consistent increase in both curves suggests effective learning. Divergence or plateauing of validation accuracy may signify overfitting.

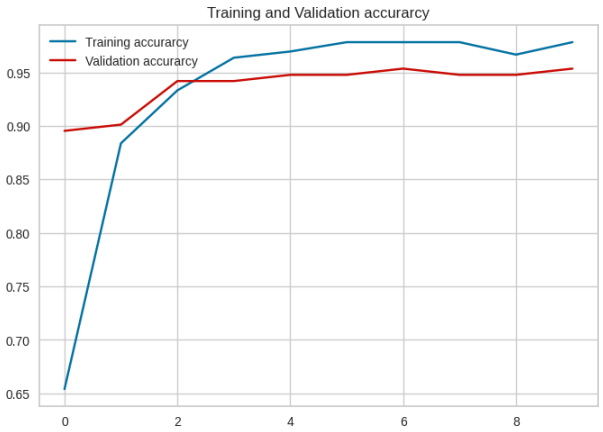

Figure 6.5.3: Accuracy Graph

In Figure 6.5.4, the plot illustrates the training and validation loss curves for a Convolutional Neural Network (CNN). The training loss, which shows how well the model fits the training data over epochs, is represented by the blue line. The validation loss, which shows how well the model performs on omitted data, is represented by the red line. A decreasing training loss indicates effective learning, while a widening gap between training and validation losses suggests potential overfitting.

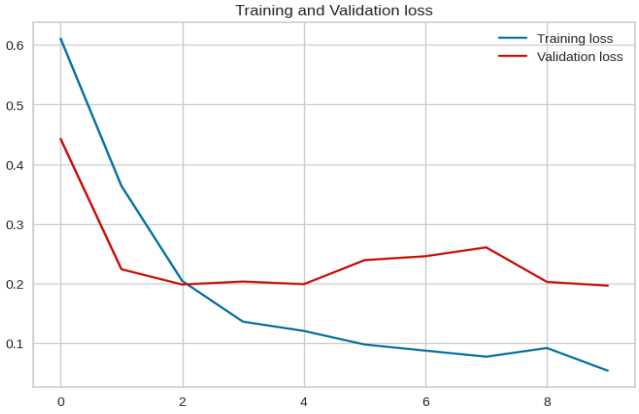

Figure 6.5.4: Loss Graph

6.6 Classification Performance of Models

In the area of Android malware detection, the classification performance of the four machine learning models, such as the Random Forest, Extra Tree, XGBoost, and Stacking Classifier, is noteworthy and one deep learning which is CNN. These classifiers regularly display high accuracy, recall, and F1-score values in both the "malign" (malicious) and "benign" (safe) classes. The "malign" class has good accuracy, recall, and F1 scores, indicating that the models can effectively detect malicious apps, with the Stacking Classifier scoring especially well. Similarly, the "benign" class has high accuracy and recall, demonstrating that the models can properly categorize benign applications.

Table 6.1: Comparison Table for "Malign" Class

Classifier	Precision	Recall	F1-Score	Support
Random Forest	92.2%	69.4%	79.2%	85%
Extra Tree	92.4%	85.9%	89%	85%
XGBoost	95.1%	90.6%	92.8%	85%
Stacking Classifier	95.2%	94.1%	94.7%	85%
CNN	96%	94%	95%	85%

Table 6.2: Comparison Table for "Benign" Class

Classifier	Precision	Recall	F1-Score	Support
Random Forest	75.9%	94.3%	84.1%	87%
Extra Tree	87.1%	93.1%	90%	87%
XGBoost	91.2%	95.4%	93.3%	87%
Stacking Classifier	94.3%	95.4%	94.9%	87%
CNN	94%	97%	95%	87%

Table 6.3: Accuracy Table of all ML and DL Models

Models	Accuracy
Random Forest	81.97%
Extra Tree	89.53%
XGBoost	92.02%
Stacking Classifier	93.76%
CNN	95.34%

7 Conclusion and Future Works

Conclusion

To efficiently detect benign and malicious Android applications, this study employed a range of machine learning classifiers, including Random Forest, Extra Tree, XGBoost, and the

Stacking Classifier and also deep learning model which is CNN. Extensive testing of these models yielded convincing results. With the highest accuracy score of 95.34 per cent, CNN stands out as the best performer, displaying its exceptional ability to correctly categorize Android applications. The models' strong classification performance, as indicated by high accuracy, recall, and F1 scores, proves their effectiveness in boosting security by recognizing potential threats and harmful programs constantly.

Limitations

While the models demonstrated impressive accuracy, several limits must be acknowledged. The dataset itself may be biased or may not reflect the entire range of Android malware, which might influence generalization. Furthermore, the features used and the size of the dataset may influence model performance. To improve model robustness, more improvements may be achieved by examining a broader variety of characteristics and using larger datasets.

Future Works

The Android malware detection system will benefit from continued study and improvement in the future. Recurrent neural networks (RNNs) may enhance feature extraction and modelling. Furthermore, regular updates to the dataset and the incorporation of real-time threat information can guarantee that the system stays current with new malware patterns. Efforts can also be put toward improving the processing and classification speeds, making it ideal for real-time mobile security applications. By resolving these issues, the Android malware detection system can improve its accuracy and dependability in protecting users from possible security threats.

References

1. Kim, T., Kang, B., Rho, M., Sezer, S. and Im, E.G., 2018. A multimodal deep learning method for android malware detection using various features. *IEEE Transactions on Information Forensics and Security*, *14*(3), pp.773-788.
2. Meijin, L., Zhiyang, F., Junfeng, W., Luyu, C., Qi, Z., Tao, Y., Yinwei, W. and Jiaxuan, G., 2022. A systematic overview of android malware detection. *Applied Artificial Intelligence*, *36*(1), p.2007327.
3. Shaukat, K., Luo, S., Varadharajan, V., Hameed, I.A. and Xu, M., 2020. A survey on machine learning techniques for cyber security in the last decade. *IEEE access*, *8*, pp.222310-222354.
4. Butt, U.A., Mehmood, M., Shah, S.B.H., Amin, R., Shaukat, M.W., Raza, S.M., Suh, D.Y. and Piran, M.J., 2020. A review of machine learning algorithms for cloud computing security. *Electronics*, *9*(9), p.1379.

5. Vanjire, S. and Lakshmi, M., 2021, September. Behavior-based malware detection system approach for mobile security using machine learning. In *2021 International Conference on Artificial Intelligence and Machine Vision (AIMV)* (pp. 1-4). IEEE.
6. Cui, L., Yang, S., Chen, F., Ming, Z., Lu, N. and Qin, J., 2018. A survey on application of machine learning for Internet of Things. *International Journal of Machine Learning and Cybernetics*, *9*, pp.1399-1417.
7. Kotenko, I., Saenko, I. and Branitskiy, A., 2018. Framework for mobile Internet of Things security monitoring based on big data processing and machine learning. *IEEE Access*, *6*, pp.72714-72723.
8. Rana, M.S., Gudla, C. and Sung, A.H., 2018, December. Evaluating machine learning models for Android malware detection: A comparison study. In *Proceedings of the 2018 VII International Conference on Network, Communication and Computing* (pp. 17-21).
9. Agrawal, P. and Trivedi, B., 2021. Machine learning classifiers for Android malware detection. In *Data Management, Analytics and Innovation: Proceedings of ICDMAI 2020, Volume 1* (pp. 311-322). Springer Singapore.
10. Koli, J.D., 2018, March. RanDroid: Android malware detection using random machine learning classifiers. In *2018 Technologies for Smart-City Energy Security and Power (ICSESP)* (pp. 1-6). IEEE.
11. Jung, J., Kim, H., Shin, D., Lee, M., Lee, H., Cho, S.J. and Suh, K., 2018, September. Android malware detection based on useful API calls and machine learning. In *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)* (pp. 175-178). IEEE.
12. Zhu, H.J., Jiang, T.H., Ma, B., You, Z.H., Shi, W.L. and Cheng, L., 2018. HEMD: a highly efficient random forest-based malware detection framework for Android. *Neural Computing and Applications*, *30*, pp.3353-3361.
13. Darus, F.M., Ahmad, N.A. and Ariffin, A.F.M., 2019, November. Android malware classification using XGBoost on data image pattern. In *2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)* (pp. 118-122). IEEE.
14. Chen, H., Du, R., Liu, Z. and Xu, H., 2018, December. Android malware classification using XGBoost based on images patterns. In *2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)* (pp. 1358-1362). IEEE.
15. Palša, J., Ádám, N., Hurtuk, J., Chovancová, E., Madoš, B., Chovanec, M. and Kocan, S., 2022. Mlmd—a malware-detecting antivirus tool based on the xgboost machine learning algorithm. *Applied Sciences*, *12*(13), p.6672.
16. Ling, J., Wang, X. and Sun, Y., 2019, April. Research of android malware detection based on aco optimized xgboost parameters approach. In *3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019)* (pp. 364-371). Atlantis Press.
17. Giannakas, F., Kouliaridis, V. and Kambourakis, G., 2022. A Closer Look at Machine Learning Effectiveness in Android Malware Detection. *Information*, *14*(1), p.2.
18. Wang, X., Zhang, L., Zhao, K., Ding, X. and Yu, M., 2022. MFDroid: A stacking ensemble learning framework for Android malware detection. *Sensors*, *22*(7), p.2597.

19. Yadav, P., Menon, N., Ravi, V., Vishvanathan, S. and Pham, T.D., 2022. A two-stage deep learning framework for image-based android malware detection and variant classification. *Computational Intelligence*, *38*(5), pp.1748-1771.
20. Xie, N., Qin, Z. and Di, X., 2023. GA-StackingMD: Android Malware Detection Method Based on Genetic Algorithm Optimized Stacking. *Applied Sciences*, *13*(4), p.2629.
21. Lee, W.Y., Saxe, J. and Harang, R., 2019. SeqDroid: Obfuscated Android malware detection using stacked convolutional and recurrent neural networks. *Deep learning applications for cyber security*, pp.197-210.
22. Shafin, S.S., Ahmed, M.M., Pranto, M.A. and Chowdhury, A., 2021, December. Detection of android malware using tree-based ensemble stacking model. In *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)* (pp. 1-6). IEEE.
23. Raczko, E. and Zagajewski, B., 2017. Comparison of support vector machine, random forest and neural network classifiers for tree species classification on airborne hyperspectral APEX images. European Journal of Remote Sensing, 50(1), pp.144-154.
24. Le, T.M., Vo, T.M., Pham, T.N. and Dao, S.V.T., 2020. A novel wrapper–based feature selection for early diabetes prediction enhanced with a metaheuristic. IEEE Access, 9, pp.7869-7884.
25. Chen, X., Wang, M. and Zhang, H., 2011. The use of classification trees for bioinformatics. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(1), pp.55-63.
26. Zaher, M., Ghoneem, A., Abdelhamid, L. and Ezzat, A., 2023. Comparative Study Between Machine learning algorithms and feature ranking techniques on UI-PRMD dataset.
27. Trizoglou, P., Liu, X. and Lin, Z., 2021. Fault detection by an ensemble framework of Extreme Gradient Boosting (XGBoost) in the operation of offshore wind turbines. Renewable Energy, 179, pp.945-962.
28. Wang, Y., Pan, Z., Zheng, J., Qian, L. and Li, M., 2019. A hybrid ensemble method for pulsar candidate classification. Astrophysics and Space Science, 364, pp.1-13.
29. Yadav, P., Menon, N., Ravi, V. and Vishvanathan, S., 2021. Lung-GANs: unsupervised representation learning for lung disease classification using chest CT and X-ray images. IEEE Transactions on Engineering Management.
30. Smmarwar, S.K., Gupta, G.P. and Kumar, S., 2022. A hybrid feature selection approach-based Android malware detection framework using machine learning techniques. In Cyber Security, Privacy and Networking: Proceedings of ICSPN 2021 (pp. 347-356). Singapore: Springer Nature Singapore.