

Enhancing Android Security: Machine Learning Approaches for App Permissions and Malware Detection

Major 2 Project Dissertation

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
SOFTWARE ENGINEERING

Submitted by

JAIKISHAN MOHANTY (2K22/SWE/07)

Under the supervision of

Dr. DIVYASHIKHA SETHIA



DEPARTMENT OF SOFTWARE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

May, 2024

DEPARTMENT OF SOFTWARE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Jaikishan Mohanty, Roll No's –2K22/SWE/07 students of M.Tech (Software Engineering), hereby declare that the project Dissertation titled “Enhancing Android Security: Machine Learning Approaches for App Permissions and Malware Detection” which is submitted by me to the Software Engineering Department, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi

Jaikishan Mohanty

Date: 24.05.2024

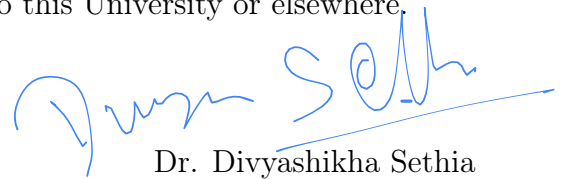
(2K22/SWE/07)

DEPARTMENT OF MECHANICAL ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled “Enhancing Android Security: Machine Learning Approaches for App Permissions and Malware Detection” which is submitted by Jaikishan Mohanty, Roll No’s – 2K22/SWE/07, Department of Software Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi



Dr. Divyashikha Sethia

Assistant Professor

Date: 01.06.2024

Department of Software Engineering, DTU

DEPARTMENT OF MECHANICAL ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

We wish to express our sincerest gratitude to Dr. Divyashikha Sethia for the continuous guidance and mentorship that she provided us during the project. She showed us the path to achieve our targets by explaining all the tasks to be done and explaining to us the importance of this project as well as its industrial relevance. She was always ready to help us and clear our doubts regarding any hurdles in this project. Without her constant support and motivation, this project would not have been successful.

Place: Delhi

Jaikishan Mohanty

Date: 24.05.2024

(2K22/SWE/07)

Abstract

The prevalence of Android devices has made the platform a major player in the mobile market. However, this widespread adoption also brings significant challenges when detecting and preventing Android malware. This thesis aims to address these challenges by utilizing machine learning techniques. The first aspect of this thesis focuses on giving users greater control over their app permissions through an innovative Android application. This application uses machine learning models to analyze the permissions requested by various apps and provides users with informed recommendations based on the safety ratings of these permissions. By leveraging usage frequency data from specific app categories on the Google Play Store, the model offers users a comprehensive tool to make educated decisions regarding app permissions, thus contributing to a safer and more secure mobile app ecosystem.

The second aspect of this thesis introduces the GARB (*Gradient Boosting classifier, AdaBoost, Random Forest and Bagging classifier with a decision tree*) Model, a novel ensemble approach for detecting malware in Android packages. The GARB Model combines multiple base classifier algorithms, including Gradient Boosting, AdaBoost, Random Forest, and Bagging Classifier with Decision Tree, through a weighted averaging method of stacking. This ensemble approach exhibits superior performance, achieving an accuracy of 82.38% in distinguishing between malware and benign Android packages. Moreover, the GARB Model outperforms individual classifiers in various criteria, highlighting its efficacy and reliability for prediction tasks related to malware detection.

Together, these contributions offer comprehensive solutions for enhancing mobile app security and user privacy within the Android ecosystem. By leveraging machine learning techniques, this thesis aims to mitigate the growing threat of Android malware and empower users with the tools and knowledge necessary to navigate the increasingly complex landscape of mobile app permissions and security risks.

Contents

Candidate’s Declaration	i
Certificate	ii
Acknowledgement	iii
Abstract	v
Content	vi
List of Tables	vii
List of Figures	viii
1 INTRODUCTION	1
1.1 Overview	1
1.2 Motivation	4
1.3 Problem Statement	4
1.4 Proposed Solution	4
1.5 Contribution	5
2 TECHNICAL BACKGROUND	6
2.1 Android System Architecture	6
2.2 An Android security Mechanism	6
2.3 Workflow for using permissions	7
2.4 Types of permissions	8
3 ML-Based Android App Permission Detection	10
3.1 Motivation And Research Gaps	10
3.2 Objective	10
3.3 Related Work	11
3.3.1 Comparison with the Existing Application	13
3.3.2 Comparison with the Android Versions	13
3.4 Methodology	14
3.4.1 Overview of APEC: App Permission Classification with Efficient Clustering [1]	14
3.4.2 Implementation of APEC in the Android Permission Detector	17
4 Stacked ML for Android App Malware Detection	20
4.1 Motivation And Research Gaps	20
4.2 Objective	20

4.3	Related Work	21
4.4	Methodology	22
4.4.1	Dataset	24
4.4.2	Pre-processing	24
4.4.3	Hyperparameter Tuning	26
4.4.4	Experimentation	26
4.5	Proposal of GARB Model	29
5	RESULTS and DISCUSSION	30
5.1	Android App Permission Detector based on Machine Learning Models . . .	30
5.2	Android App Malware Detection using Stacking of Machine Learning Algorithms	31
6	CONCLUSION AND FUTURE SCOPE	34

List of Tables

3.1	Related Work On Model	12
3.2	Comparison of Android	14
3.3	Comparison with Android Versions	16
4.1	Related Work	24
5.1	Application Testing with Proposed Application	30
5.2	Performance Comparison of Different Algorithms	32

List of Figures

2.1	Platform Architecture of Android [2]	7
2.2	Permission-Based Security Mechanism	8
2.3	Illustrates the workflow for using app permissions [3]	8
3.1	Flow diagram of Three-Tier Architecture for APEC [1]	15
3.2	K-distance graph of Action category [1]	17
3.3	Flow diagram of Android Permission Detector	18
4.1	Flow Diagram	23
4.2	Dataset Distribution	25
4.3	Price vs Category of applications	26
4.4	Different Experimental Phases	27
4.5	GARB (<i>Gradient Boosting classifier, AdaBoost, Random Forest and Bagging classifier with a decision tree</i>) Model	29
5.1	The interface of Permission Detector. In Fig. (a), Audiobooks show granted unsafe permission, of which three are user-authorized. In Fig. (b), After turning off User-authorized permission from setting and refresh the app.	31
5.2	Confusion Matrix of GARB Model	32

Chapter 1

INTRODUCTION

1.1 Overview

In recent years, the Android ecosystem has experienced exponential growth, emerging as the dominant player in the global mobile operating system landscape [4]. With a staggering market share of approximately 84.2% worldwide and over 3 billion active users, Android's accessibility and widespread adoption have reshaped the way individuals interact with technology [5]. However, amidst this meteoric rise, the platform has encountered a formidable adversary: malware.

Malware, or malicious software, poses a significant threat to the privacy and security of Android users and has become a pressing concern for both consumers and developers alike [6]. The proliferation of malware on the Android platform has escalated to alarming levels, with an overwhelming majority—nearly 97% of malicious mobile apps targeting Android devices [7]. This threat manifests in various forms, ranging from seemingly innocuous applications masquerading as ad blockers to sophisticated malware capable of infiltrating devices and accessing sensitive data [8].

The Android security mechanism, while robust in theory, has exhibited vulnerabilities in practice, contributing to the proliferation of malware [6]. The app publishing process, characterized by a lack of stringent validation, has provided an entry point for malware publishers to exploit, circumventing security measures and infiltrating the platform [6]. Consequently, Android users are left vulnerable to many privacy breaches and data abuses, jeopardizing their personal information and digital identities [9].

Android presents several privacy threats [10], such as:

- (a) *Malware*: Without the user's knowledge, malicious software or malware is installed on an Android device, gathering private data, including login passwords and financial information.
- (b) *Location tracking*: A few applications can access and track a user's location without that user's knowledge or consent.
- (c) *Data Leakage*: Android apps can access various private data, including contacts, location, and text messages, and they may do so without the user's knowledge or approval.

- (d) *Phishing*: Phishing attacks target Android users, tricking them into providing sensitive information, including login credentials, to a bogus website or app.

Applications must obtain user permission before accessing specific data and capabilities under Android's permission system. However, it is ultimately up to the users to allow or deny these permissions, which is problematic. More than 70% smartphone apps often request data collection permissions that are only sometimes essential for their core functions. Unfortunately, many users hastily grant all permissions during installation, while even those who pay attention may need help comprehending the implications fully. The lack of practical guidance for granting permissions leaves users needing assistance navigating the minefield of app permissions. The consequences are dire, as third-party apps can covertly track user locations, access sensitive information like contact lists and SMS logs, and compromise personal data privacy [9]. With Android 6.0, Android introduced runtime permissions to address these issues and give users more control over app permissions. There are four types of permissions: install-time permissions, runtime permissions, normal permissions, and special permissions. These permissions define the scope of data access for an app and the actions the app can perform if granted permission.

1. *Install-time permission*: It provides restricted access to classified data or actions that have a minimal impact on the system. The user receives a notification when they view the app's information page, notifying them of the install-time permissions. Examples of these permissions include ACCESS_NETWORK_STATE, BLUETOOTH, and INTERNET.
2. *Runtime permission*: It grants the app additional access to classified data or actions that can significantly impact the system and other apps. Therefore, the app must request runtime permissions before accessing classified data or performing restricted actions. Examples of these permission groups include Calendar, Call log, Camera, Contacts, Location, Microphone, Sensor, SMS, Storage, and Telephone.
3. *Special permission*: These permissions are established when the platform and Original Equipment Manufacturers(OEMs) wish to restrict access to decisive actions such as drawing over other apps. The Special app access page in system settings holds a collection of user-toggle-able operations, with many implemented as special permissions. These permissions include scheduling exact alarms, drawing over other apps, and Accessing all storage data.

Runtime permissions are also known as user authorization permissions, primarily encompassing permission groups. However, none of the Android versions provide any service to warn the user about the safety of the application permissions. The app permissions management system has changed since the recent Android 12 update [11]. The update's notification service covers only a small number of specific permissions. Other permissions may also pose a threat to privacy. New malware techniques developed for new apps mainly threaten users' privacy. Therefore, there is a need for new Android services that can warn users when something is safe or unsafe and give them a secure environment in which to use applications.

Android 13 [12] brings several vital updates. Users can now stop foreground services from the notification drawer, improving performance and battery life. A new runtime permission, POST_NOTIFICATIONS, enhances user control over notifications. Apps targeting Android 13 must declare the AD_ID permission for Google Play services. Privacy and security need separate media access permissions instead of READ_EXTERNAL_STORAGE.

Install-time permission `USE_EXACT_ALARM` benefits apps like calendars and alarms, ensuring precise timing. These changes reflect Android’s Focus on performance, privacy, and security.

To address this problem, APEC, also known as App Permission Classification with Efficient Clustering [1], is a model that aims to enhance the security of Android applications. It utilizes a three-tier structure consisting of permission categorization clustering through DBSCAN and permission classification using Decision Tree and Random Forest algorithms. APEC achieves accuracy rates, with Decision Tree reaching 93.8% accuracy and Random Forest achieving 95.8% accuracy. Moreover, it effectively addresses privacy concerns by predicting whether permissions are safe or unsafe based on the categories of the apps. This model offers recommendations for users and developers, improving Android app security in line with the updates introduced in Android 12 [11] for permission management.

According to previous research, such as the study conducted by Wang et al. [13], machine learning can aid in the identification of Android malware. However, prior efforts have focused mainly on detecting malware, with little attention paid to privacy concerns arising from the apps’ additional permission requests. Therefore, a new approach is necessary to address the privacy and data abuse issues that may arise from the apps’ permission requests.

One approach to mitigating the malware threat on Android devices is using the Android permission system. This system mandates that applications acquire user consent before accessing specific data and capabilities [9]. However, studies have revealed that users often grant permissions without fully comprehending their implications, inadvertently exposing themselves to privacy risks [9]. Furthermore, while machine learning techniques have shown promise in detecting Android malware [13], existing efforts have primarily focused on malware detection, neglecting the associated privacy concerns arising from excessive permission requests [13].

This thesis endeavours to address the multifaceted challenge posed by malware on the Android platform, with a particular emphasis on enhancing malware detection while safeguarding user privacy. Leveraging datasets such as the Android Permission Dataset (AP dataset) [14], this research aims to develop a comprehensive approach to malware detection based on permissions. Building upon previous studies, such as the XGBoost-based model proposed by Rawat et al. [15], this thesis seeks to refine and augment existing methodologies by adopting ensemble learning techniques.

By amalgamating insights from diverse domains, including cybersecurity, machine learning, and mobile computing, this thesis aims to contribute to developing more secure and privacy-aware Android applications. The overarching goal is to foster a safer digital ecosystem for Android users, characterized by heightened resilience against malware threats and enhanced personal data protection. Through empirical research and practical interventions, this thesis seeks to fortify the foundations of trust upon which the Android ecosystem thrives, ensuring that users can confidently harness the power of technology without compromising their privacy or security.

1.2 Motivation

The exponential growth of the Android ecosystem has brought along significant security challenges, particularly in combating malware threats. With Android devices being the target of a vast majority of malicious mobile apps, there's an urgent need for robust cybersecurity measures. Despite efforts to address this issue, vulnerabilities in the Android security mechanism and app publishing process persist, allowing malware to infiltrate the platform.

The motivation for this research stems from the critical gaps in current malware detection techniques, particularly in addressing privacy concerns related to excessive permission requests. By leveraging datasets like the Android Permission Dataset (AP dataset) and employing advanced machine learning techniques, this research aims to enhance malware detection accuracy while minimizing privacy risks for Android users.

Ultimately, the goal is to contribute to the development of more secure and privacy-aware Android applications, fostering a safer digital ecosystem where users can confidently utilize technology without compromising their privacy or security.

1.3 Problem Statement

In today's data-driven world, security has become a paramount concern due to the massive volume of data generated daily. One practical approach to addressing this challenge is identifying potentially harmful behaviour of Android applications as their usage proliferates. This can be achieved through dynamic malware analysis techniques based on Android permission analysis. Utilizing machine learning algorithms such as Decision Trees and Random Forests, this research aims to detect and classify application permissions as safe or unsafe, enhancing Android app security.

Another significant problem is malware detection on Android devices. While existing methods have shown promise, there remains a need for more robust solutions that can accurately identify and mitigate malware threats. This research uses an ensemble approach to enhance malware detection accuracy by combining multiple classifiers, thereby fortifying Android device security against evolving malware threats.

1.4 Proposed Solution

Building upon the aforementioned contributions, the proposed solution for this thesis report involves developing and implementing a comprehensive framework for Android app security.

The framework encompasses two primary components:

1. *Android App Permission Detector*: The first component involves designing and deploying the Android App Permission Detector. Leveraging the insights gained from the APEC model, the detector will analyze and classify application permissions to determine their safety levels. By integrating with the latest Android updates and utilizing advanced machine learning techniques, the detector will provide actionable recommendations for developers to optimize app permissions, enhancing overall app security.

2. *GARB Model for Malware Detection:* The second component focuses on implementing the GARB Model for malware detection. By utilizing ensemble learning techniques and a diverse set of machine learning algorithms, the GARB Model will accurately identify malicious Android packages. By leveraging the strengths of individual classifiers and employing stacking with Logistic Regression, the model will enhance malware detection accuracy while minimizing false positives.

The proposed solution combines these two components within a unified framework to provide a comprehensive approach to Android app security. Through empirical evaluation and real-world testing, the effectiveness and robustness of the solution will be assessed, with the ultimate goal of enhancing user privacy and security in the Android ecosystem.

1.5 Contribution

The proposed work makes significant contributions in two key areas:

1. **Proposal of Android App Permission Detector:** This research introduces a novel Android App Permission Detector designed for permission classification based on a static model known as APEC [1]. The detector evaluates the safety of application permissions by analyzing their frequency of usage within specific app categories on the Google Play Store. Aligned with the updates introduced in Android 12 and Android 13 [11, 12], the detector outperforms similar applications in the market. Furthermore, it serves as a user recommendation system, providing developers with insights into the minimal permissions required for optimal app functionality while alerting users about potentially unsafe permissions associated with installed applications.
2. **Proposal of GARB Model for Malware Detection:** This study proposes the innovative GARB Model, which stands for Gradient Boosting, AdaBoost, Random Forest, and Bagging Classifier with Decision Tree. Utilizing a weighted averaging method of stacking, the GARB Model determines whether an Android package is malware. By employing various machine learning algorithms, including Gradient Boosting, Ridge Classifier, Random Forest Classifier, MLP Classifier, Bagging Classifier with Decision Tree, and ADA Boost Classifier, this research explores ensemble prediction techniques to enhance malware detection accuracy. Additionally, stacking with Logistic Regression is employed, utilizing base models' predictions to train a meta-model for improved malware identification.

Chapter 2

TECHNICAL BACKGROUND

2.1 Android System Architecture

The Android operating system, a Linux-based kernel system, uses a software Stack to create its hierarchical system architecture. According to Figure 2.1, which is ordered from bottom to top are as follows- Power Management, Android Runtime environment, hardware abstraction layer (HAL), Linux kernel, native C/C++ libraries, Java API framework, an application layer, Google offers the traditional layered design of the Android system. There are numerous subsystems and submodules within each layer. The Linux kernel is the foundation of the kernel section at the bottom of the Android stack, as shown in Fig. 3.1.1. While the native C/C++ libraries, Android Runtime Environment, and the Java API framework make up the user space at the top of the system. System calls connect the kernel and user areas. Most user space programs are created using Java or C++. Through the Java native interfaces, the user space's Java layer and native layer are linked to the rest of the Android system.

2.2 An Android security Mechanism

Android OS is a privileged-partition operating system in general. A series of system services are carried out by the system using Binder, an inter-process communication mechanism to achieve high-level system operations as shown in figure 2.2. Utilizing their distinct system identifiers, the Android operating system isolates executing applications (Linux UIDs). By default, Android applications are only given a small number of permissions in order to communicate with hardware, system services, and other applications, they must acquire more specific permission. An Android application's required permission is listed in the relevant manifest file, i.e. `AndroidManifest.xml`, and is provided either after installation or while the program is in a running state. The Android operating system categorizes the permissions given to each program using UIDs, executes these restrictions while the program is running and further limits each process' permissions by using SELINUX.

The development of security features has received a lot of focus from Android developers during system updates and iterations. For instance, the 2019 release of Android Q includes a number of additional security features, including access control for sensitive data, file-based encryption, access control for background cameras and microphones, encrypted backup, a lock mode, and a system known as Google Play Protect. Android version

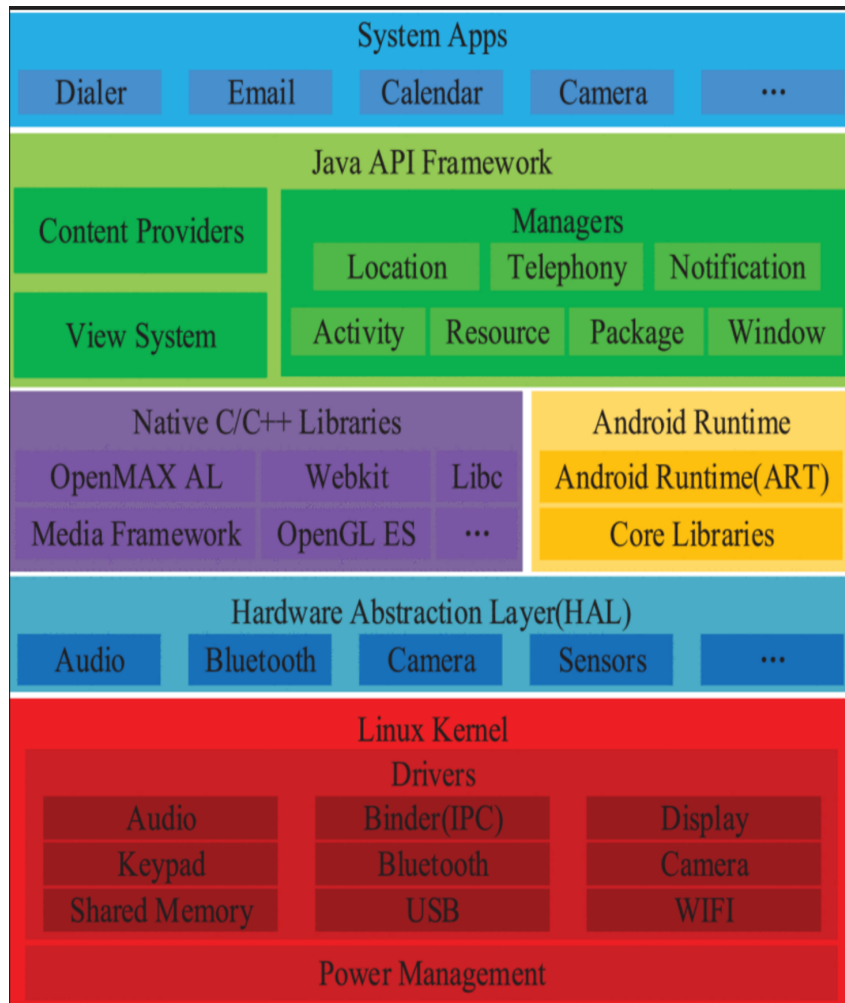


Figure 2.1: Platform Architecture of Android [2]

Q offers a variety of security and privacy protections for users. Additionally, it has a better system for controlling permissions, gives the users more control over sharing their location, prevents background programs from initiating tasks, limits app. access to the non-reset device identifiers (like the serial number and IMEI), and by default, it enables MAC address randomization. Android Malware is still a concern, though.

2.3 Workflow for using permissions

In the event that your app necessitates access to classified data or actions, it is important to establish whether or not you can obtain the information or perform the actions without announcing your permissions. There are many use cases in your application, such as capturing images, stopping media playback, and presenting related adverts, that can be accomplished without announcing any permissions. However, if it is determined that the app must obtain classified data or execute restricted actions to achieve its objectives, then the necessary permissions must be declared. Certain permissions, known as install time permissions, are granted automatically upon app installation. Other permissions, known as runtime permissions, require the app to request permission at runtime

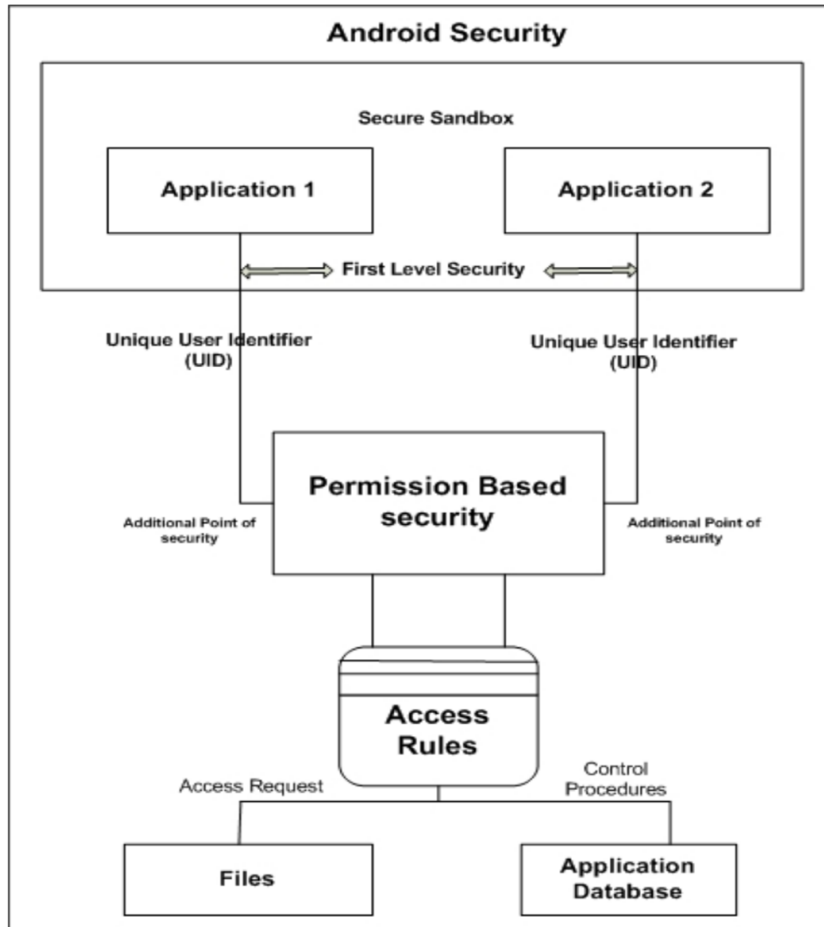


Figure 2.2: Permission-Based Security Mechanism



Figure 2.3: Illustrates the workflow for using app permissions [3]

2.4 Types of permissions

There are various types of permissions categorized by Android, including install-time permissions, runtime permissions, and special permissions. Each permission type specifies the range of classified data that your app can access and the range of restricted actions that your app can perform if your app is granted that permission.

1. *Install-time permissions*: Install-time permissions offer limited access to classified data or allow the app to execute restricted actions that have a minor impact on the system or other apps. When you include install-time permissions in your app, the app store displays an install-time permission notification to the user when they view the app's information page.
2. *Normal permissions*: These permissions provide access to data and actions that go beyond your app's sandbox but pose a minimal risk to the user's privacy and the operation of other apps. Normal permissions are assigned the normal protection level by the system.
3. *Signature permissions*: The system grants signature permission to an app only when the app is signed by the same certificate as the app or the OS that defines the permission. Apps that implement privileged services, such as autofill or VPN services, also require signature permissions for service binding, so that only the system can bind to the services. The signature protection level is assigned to signature permissions by the system.
4. *Runtime permissions*: Runtime permissions, also known as dangerous permissions, provide the app with additional access to classified data or allow the app to execute restricted actions that significantly impact the system and other apps. As a result, runtime permissions must be requested in the app before accessing classified data or performing restricted actions. Don't assume that these permissions have been granted in advance, instead check them and, if necessary, request them before each access.
5. *Special permissions*: Special permissions correspond to specific app operations. Only the platform and OEMs can define special permissions. Additionally, the platform and OEMs typically establish special permissions when they wish to restrict access to particularly powerful actions, such as drawing over other apps. The Special app access page in system settings contains a set of user-toggable operations. Many of these operations are implemented as special permissions.

Chapter 3

ML-Based Android App Permission Detection

3.1 Motivation And Research Gaps

The motivation behind this research lies in the persistent challenges surrounding Android app security and privacy. Despite advancements in mobile technology, users still need help understanding and managing app permissions effectively. The dynamic nature of malware threats further complicates the landscape, necessitating adaptive solutions that can keep pace with evolving risks.

Moreover, while machine learning holds promise for improving security measures, there still needs to be a gap in seamlessly integrating these advanced techniques into practical applications. Existing tools may require more user-friendliness, hindering their adoption and effectiveness.

In response to these challenges, this thesis aims to develop an intuitive Android application that leverages machine learning models for permission classification. By providing users with actionable insights and recommendations, the application seeks to empower users to make informed decisions about app permissions, ultimately contributing to a safer and more secure mobile app ecosystem.

3.2 Objective

The objective of this thesis is to develop an Android application utilizing machine learning models to enhance the security and privacy of Android users by providing a comprehensive tool for assessing app permissions. The application aims to address the critical issue of Android malware detection and privacy breaches resulting from the lack of stringent validation during app publishing. Specifically, the objectives are:

- (a) To develop an Android application that employs machine learning models to assess the safety of permissions requested by various apps available on the Google Play Store.
- (b) To integrate the machine learning model into the Android application to classify app permissions as safe or unsafe based on their usage frequency in specific app categories.
- (c) To design a user-friendly interface that empowers users to make informed decisions

regarding app permissions by providing safety ratings and recommendations.

- (d) To contribute to a safer mobile app ecosystem by providing users with practical guidance for managing app permissions and mitigating potential security and privacy risks.

3.3 Related Work

In 2022, Manzil et al. [16] proposed a detection framework based on permission features using machine learning techniques and Recursive Feature Elimination (RFE) technology. The system analyzes 100 CoVID-themed fake apps from the Google Play Store and GitHub repository and extracts permission features from the AndroidManifest.xml file. The system consists of 4 components: dataset collection, static analysis, feature selection, and classification. The study shows better accuracy with the Decision tree and random forest classifiers. Most malware identified are part of the Cerberus, SpyNote, Metasploit payloads, and SMS Stealer family. This framework may not detect new or unknown variants of COVID-themed Android malware that researchers have yet to include in the dataset.

SAMADroid, a novel three-level hybrid malware detection model for the Android operating system, was proposed by Arshad et al. (2018) [17]. The authors conduct an analysis and categorize several Android malware detection methods. They developed a revolutionary three-layer hybrid malware detection methodology for the Android operating system that combines the advantages of three layers: static and dynamic analysis, local and remote hosts, and machine learning intelligence. According to trial data, SAMADroid ensures efficiency in terms of power and storage consumption to reach a high level of accuracy.

Sun et al. (2016) [18] proposed a method called Significant Permission Identification for Android Malware Detection (SIGPID). This method aims to tackle the malware issue on Android devices by identifying the required crucial permissions. Their study showed that SIGPID outperformed techniques in terms of effectiveness. It achieved an accuracy rate of 93.62% for detecting apps. It has demonstrated a classification accuracy of 91.4% when tested with unknown malware apps.

Multilevel Permission Extraction (MPE), a method to automatically detect permission interactions that successfully discriminate malicious and benign applications, was proposed by Zhen Wang et al. in 2019 [19]. It uses three techniques for extracting permissions: association rule mining, PCA, and deep cross-network (DCN). MPE achieves a malware detection rate of 97.88% by classifying apps using machine learning algorithms. The experiment’s findings demonstrate that MPE can attain accuracy, recall, precision, and FScore levels above 95.8%. In the two cutting-edge methods compared to MI and SigPID, MPE achieves 96.58% and 95.69% detection rates.

S. Alsoghyer et al. in 2020 [20] proposed a ransomware threat on Android devices by analyzing Android permissions and proposed a permissions-based detection system. The study uses a 500 ransomware and 500 benign apps dataset, finding distinct permission patterns. The system achieves a 96.9% accuracy rate, particularly with Random Forest. This research emphasizes the importance of Android permissions in ransomware detection, provides a precise model, and shares the dataset for further study, offering an efficient approach to enhance Android security against ransomware.

Table 4.1 contrasts the related work’s main contributions and limitations. The earlier

Table 3.1: Related Work On Model

Research Work	Type	Analysis Tech. Used	Contribution	Limitation
Manzil et al. (2022) [16]	Malware Detection	Static Analysis	It identifies COVID-themed Android malware and improves accuracy using Recursive Feature Elimination and machine learning.	The dataset was custom-created using feature elimination techniques and may not detect new or unknown variants of COVID-themed Android malware not in the dataset.
Sun et al., (2017) [18]	Malware Detection	Static Analysis	Multilevel Data Pruning	Out of 122 possible permissions, the model only takes 23 into account. False classification may be the outcome of the reduction.
Arshad et al. (2018) [17]	Malware Detection	Static and Dynamic Analysis	SAMADroid have shown a slight performance overhead on Android smartphones, outperforming the Drebin dataset in static analysis.	The local host does not detect malicious activity, resulting in the failure of network links or congestion at the channel.
Zhen Wang et. Al (2019) [19]	Malware Detection	Static Analysis	Multilevel Permission Extraction for Malware Detection.	Tested on a specific dataset of 4,868 benign and 4,868 malicious applications.
S. Al-soghyer (2020) [20]	Malware Detection	Hybrid (Static and Dynamic analysis)	Permissions-based ransomware detection system for Android devices achieved 96.9% accuracy and offers valuable insights into the critical role of Android permissions in ransomware detection.	Worked on 500 ransomware and 500 benign Android apps without real-world testing.

studies demonstrate that the dataset needed to be revised and unjustifiable given the variety of apps available in the Google Play Store. Additionally, the prior efforts' main emphasis was on malware identification rather than the privacy issues brought on by the programs' additional permission requests. Therefore, a new approach is needed to solve the privacy and data abuse issues brought on by the apps' other permission requests.

3.3.1 Comparison with the Existing Application

Mobile app permissions are crucial for user security and privacy. Applications have evolved to manage permissions, allowing users to turn them on or off. Malware exploiting app permissions is a growing threat to user privacy.

This paper delves into a comparative analysis of the "Permission Detector" application and several similar applications in the domain of permission management. It provides a comprehensive view of the functionality, limitations, and malware detection capabilities of Permission Detector compared to other permission management applications. This comparative analysis aims to highlight the distinctive strengths and contributions of Permission Detector in enhancing mobile app security and user privacy.

The Permission Manager [21] primarily focuses on enabling or disabling specific permissions for user-installed and system applications. However, it operates with limited permissions, including Calendar, Call log, Camera, Contacts, Location, Microphone, Sensor, SMS, Storage, and Telephone. Notably, it cannot modify sensor permissions starting from Android 12. Despite its functionality, Permission Manager does not address malware detection through permission classification.

Similarly, Permission Pilot [22] presents a dashboard listing all installed applications and their permissions from the Manifest.xml file. Users have control over viewing and disabling granted and ungranted permissions, but like Permission Manager, it lacks malware detection capabilities based on permission classification.

On the other hand, App Permission Manager [23] provides information about installed, system, and high-risk apps, specifying granted permissions from various categories. While identifying high-risk apps and considering granted permissions without evaluating their safety. Like the other tools, App Permission Manager does not feature malware detection based on permission classification.

Table 3.2 describes that other applications provide valuable functionality for managing app permissions. The proposed Android application Permission Detector stands out with its unique capability for malware detection through permission classification. It empowers users to not only control permissions but also assess the safety of these permissions, contributing to a more secure and privacy-conscious mobile app ecosystem.

3.3.2 Comparison with the Android Versions

Android released a new update with significant changes in the app permission management. Here is a comparison of updates in the app permission management with the proposed application as shown in Table 3.3.

Table 3.2: Comparison of Android

Aspects	Number of permissions	Purpose	Safe/Unsafe support
<i>Permission Manager [21]</i>	Only user authorizes permissions	Enable and Disable permission.	No
<i>Permission Pilot [22]</i>	All permissions of AndroidManifest.xml file.	Display Granted and ungranted permissions, and the user can disable it from the setting.	No
<i>App Permission Manager [23]</i>	Only user authorizes permissions	Presents high-risk apps based on four granted permissions: Contact, SMS, Telephone, and Storage.	No
<i>Proposed Android Permission Detector</i>	All Runtime and install time granted permissions.	Detect safe or unsafe permission based on their usage frequency in the app category.	Yes

3.4 Methodology

3.4.1 Overview of APEC: App Permission Classification with Efficient Clustering [1]

The APEC model [1] is an approach for classifying app permissions into safe and unsafe categories using a three-tier architecture. It uses a 2 million dataset comprising the name of the app and all app permissions requested by the applications. It efficiently categorizes app permissions based on the frequency of their occurrence within different app categories. The key components of the APEC model, as shown in Fig. 3.1, comprise of Group category, Clustering and Approval and Classifier.

1. *Group Category*: The APEC model categorizes apps based on their respective categories in Layer 1, as shown in Fig. 3.1. This categorization stems from apps within the same category generally sharing similar requirements and permissions, as explained in [24]. To group apps by category, the APEC model calculates the sum of each unique permission in every category on the basis requested by the apps. It then creates a frequency map of permissions based on the category containing the sum of each permission.
2. *Clustering and Approval*: The second tier of the APEC model uses DBSCAN clustering to group permissions based on their frequency in Layer 2, as shown in 3.1. DBSCAN is a density-based spatial clustering of applications with a noise algorithm well-suited for clustering data with outliers. The APEC model uses DBSCAN clustering to group permissions into core and outlier clusters. The core cluster contains the permissions most frequently requested by apps in a category, while the outlier cluster contains the less frequently requested permissions.

To determine which permissions belong to the core cluster, the APEC model calculates the k-distance for each permission. The k-distance is the minimum distance

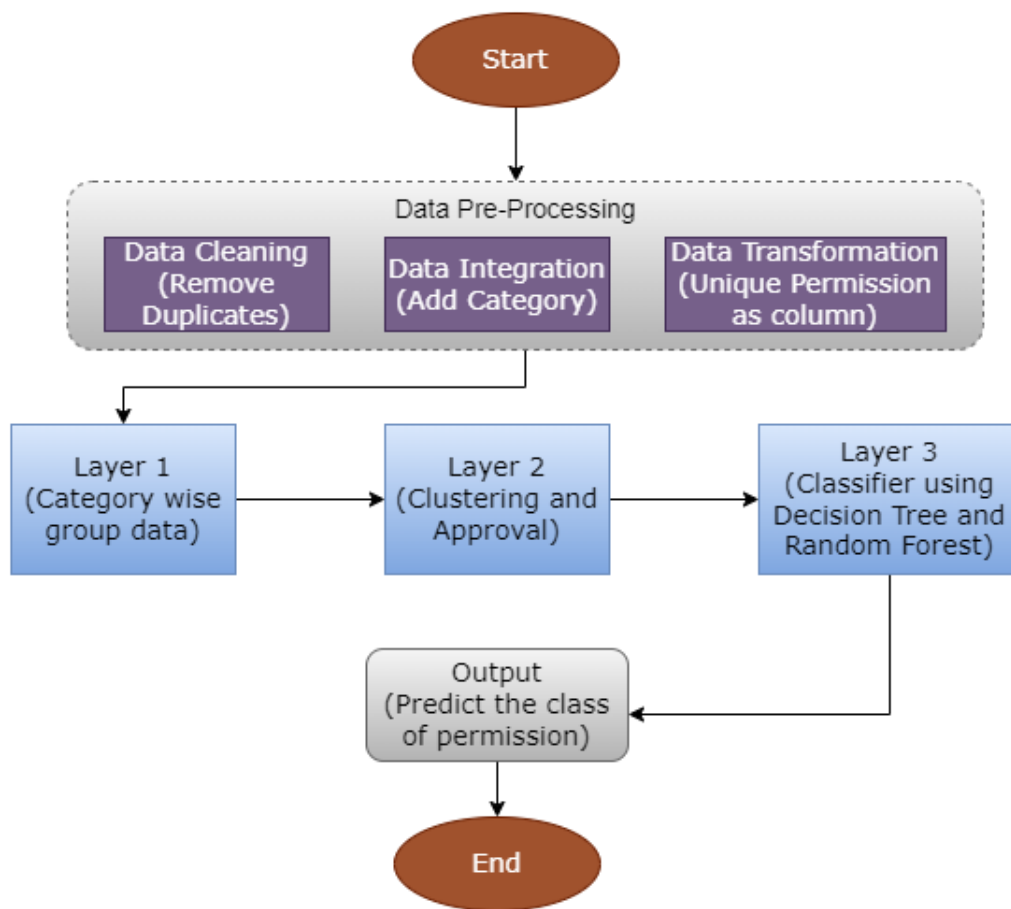


Figure 3.1: Flow diagram of Three-Tier Architecture for APEC [1]

Table 3.3: Comparison with Android Versions

Aspects of Comparison	Selective Permission Control	Scope of Permission Types	Privacy Concerns	Comprehensive Coverage
<i>Android 12 [11]</i>	Selective notification for microphone and camera access only.	Focus on platform-provided permissions, excluding runtime and special permissions.	Permission groups might only address some privacy concerns and variations among apps.	Limited scope, concentrating on a few specific permissions.
<i>Android 13 [12]</i>	User can stop Foreground service from the Notification drawer.	Some of the runtime permissions covered, e.g., POST_NOTIFICATION	Separate permission for EXTERNAL_STORAGE	Calendar and Alarm introduce some new permissions, i.e., USE_EXACT_ALARM at install time.
<i>Proposed Permission Detector</i>	Notifications cover all user permissions.	Consider a wide array of permissions, including runtime and special ones.	Permission categories ensure better addressing of various privacy concerns.	It covers a wide range of permissions to provide comprehensive coverage.

between permission and its k th nearest neighbour. The APEC model then plots the graph of k -distance values and identifies the point where the line curves. Fig. 3.2 shows the k -distance graph of the action category. It shows the line approximately curves at a value of $\varepsilon = 20$. Hence, in DBSCAN clustering, the value will be in the neighbourhood of 20 to give an optimal clustering of the app permissions of the action category. This point is considered the optimal epsilon value, and the cluster uses it for evaluation.

The APEC model also assigns an approval rating to each permission. The approval rating is represented by a '1' for the cluster's core, showing that the permission is safe, and an unsafe '0' for the cluster's outlier, showing each category's rare permissions requests.

3. *Classifier*: The third tier of the APEC model [1] uses a decision tree or random forest classifier to evaluate the permissions of new apps according to their category and the approval rating of the permissions within that category. Tier 2 clustering generates ground truth data and trains the decision tree or random forest classifier. To evaluate the permissions of a developed application, the APEC model follows a step-by-step process. Firstly, it identifies the category that the app belongs to. Then, it assigns an approval rating for each permission associated with that category.

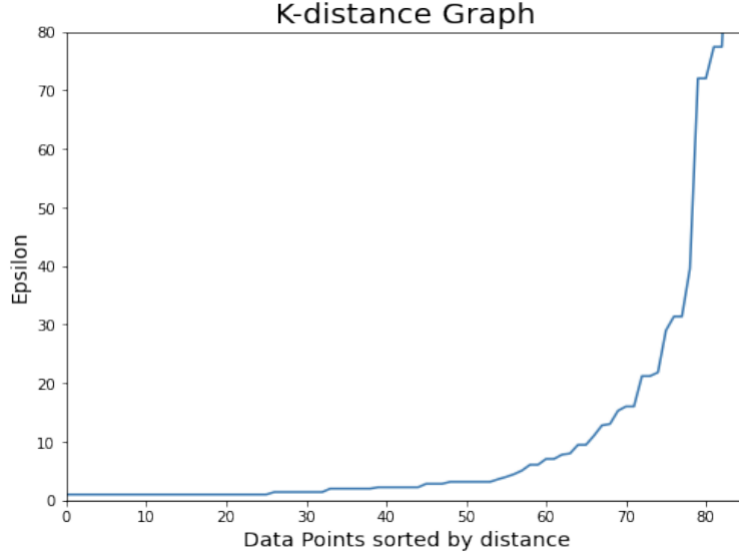


Figure 3.2: K-distance graph of Action category [1]

Lastly, using either a decision tree or random forest classifier, it predicts the safety level of each permission by considering its approval rating.

3.4.2 Implementation of APEC in the Android Permission Detector

An Android application that implements the APEC model [1] allows users and developers to assess the security of app permissions. The application provides a user-friendly interface to assess the permissions requested by various apps and recommends the appropriate actions based on the permissions classification.

Integrating the APEC model [1] into the Android application is seamless, providing users with a straightforward and user-friendly experience. This integration allows users to access the APEC model's functionality without complexity. The integration of the APEC model [1] into the Android application, as shown in Fig. 3.3, involves several key steps:

1. *Dataset*: The work enhances the dataset used in the APEC [1] Model by extracting the model predicted output and integrating it into the Android application known as Permission detector.
2. *Initial Installation and Data Collection*: Upon the initial installation of the application, it queries all application packages and Loaded APEC model results into an application.
3. *Pre-processing Dataset*: The collected dataset of query packages undergoes pre-processing to eliminate all system applications in the list. This step ensures that the analysis focuses on user-installed applications only.
4. *Exporting Permissions*: Users can export all user-installed apps and their granted permissions into an Excel file. This functionality supports future permission classification, data analysis, and app security work.
5. *Fetching Granted Permissions*: When the user clicks on any application, it fetches all

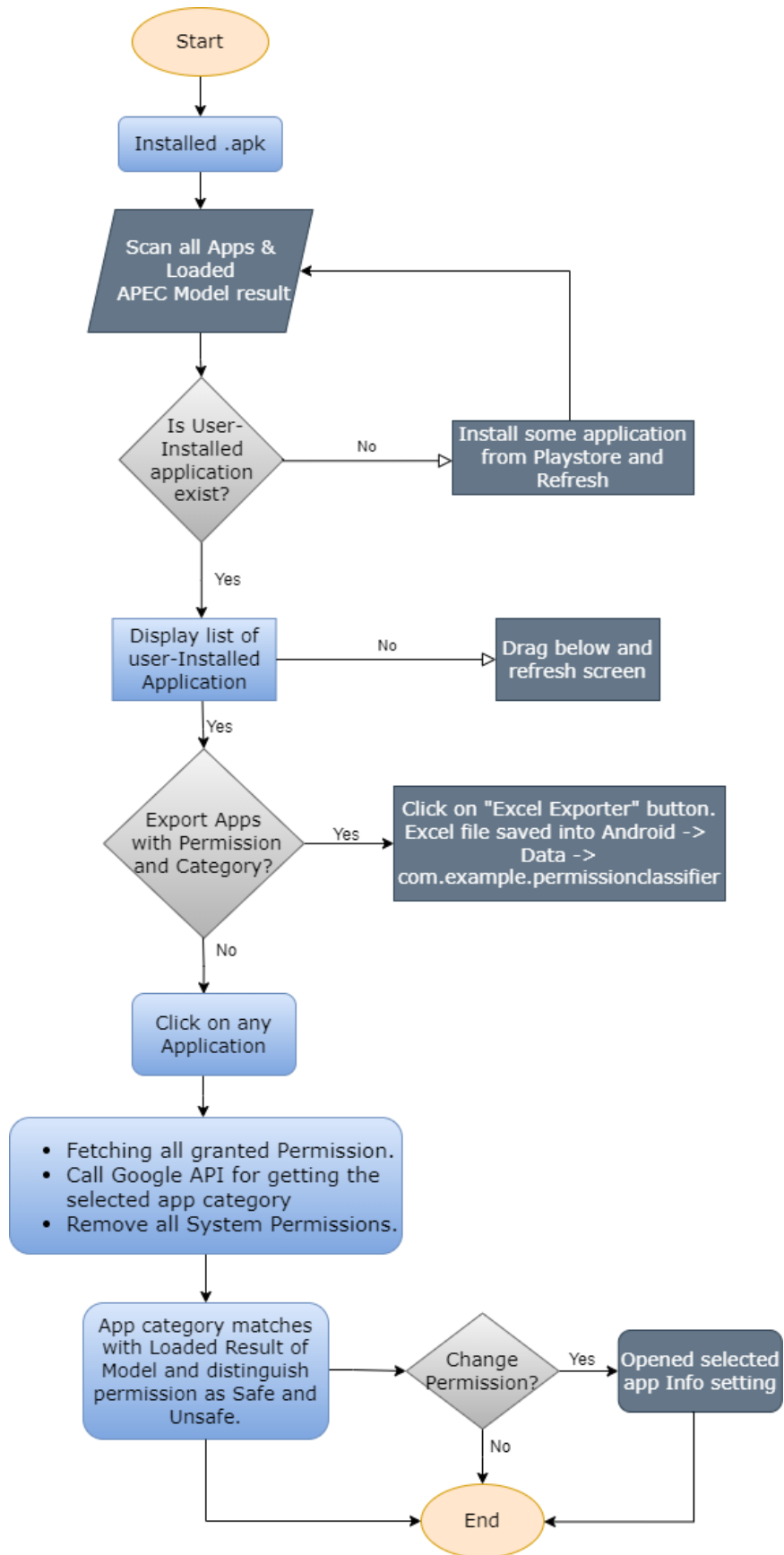


Figure 3.3: Flow diagram of Android Permission Detector

the granted permissions by checking if `PackageInfo.requestedPermission == PackageManager.PERMISSION_GRANTED`, then add in a list, forming the basis for further analysis.

6. *Comparison with APEC Model*: Compared the granted permissions with the APEC model's [1] results. This comparison categorizes permissions as safe or unsafe, providing users with clear insights into their security.
7. *Enable or Disable Permission*: Users can interact with the application through a user-friendly interface. They can turn specific permissions on or off directly from the application, giving them greater control over their device's security.

Chapter 4

Stacked ML for Android App Malware Detection

4.1 Motivation And Research Gaps

The increase in malware attacks on mobile devices, notably Android smartphones, highlights the urgent need for more effective detection methods. Despite efforts, current techniques often struggle to accurately identify malware, leaving users vulnerable to privacy breaches and security threats. While machine learning algorithms offer promise, further advancements are needed to improve detection accuracy and address evolving malware tactics. Additionally, there is a need to explore the privacy implications of application permissions, an area often overlooked in current research.

Existing literature lacks enhanced detection accuracy, privacy considerations, exploration of ensemble learning, and comprehensive model comparisons. Despite leveraging machine learning, current methods still need the desired accuracy. This research proposed an ensemble approach, the GARB Model, which addresses this gap by combining multiple classifiers. Previous studies focus primarily on malware detection, neglecting the privacy risks of excessive application permissions. This research seeks to bridge this gap by examining the broader implications of permission requests. Ensemble techniques, particularly in mobile malware detection, still need to be explored. This study investigates the effectiveness of combining classifiers to improve detection performance. It is essential to conduct more comprehensive comparisons between machine learning algorithms and ensemble methods for detecting malware on Android devices. This thesis work performs a thorough evaluation to identify the most effective approach.

4.2 Objective

The objective of this paper is to investigate the effectiveness of various classification algorithms in identifying malware in Android applications based on the permissions requested by the user. The study aims to determine the most accurate model for predicting malware presence in different types of Android packages. Specifically, the objectives are:

- (a) To evaluate the performance of different classification algorithms, including Gradient Boosting, AdaBoost, Random Forest, and Bagging Classifier with Decision Tree (GARB), in identifying malware in Android applications using permission-based

features.

- (b) To implement a weighted averaging ensemble approach, incorporating multiple classification models, to enhance the accuracy of malware detection in Android packages.
- (c) To propose the GARB model, leveraging the strengths of various classification algorithms through ensemble prediction, and compare its performance with individual classifiers and other ensemble methods.
- (d) To explore the efficacy of stacking techniques, particularly using Logistic Regression as a meta-model, in improving the accuracy of malware detection based on permission requests in Android applications.

4.3 Related Work

Rawat et al. [15] presents a comprehensive investigation into machine learning algorithms for Android malware detection, focusing on permissions requested by apps. Previous research has predominantly utilized deep learning and supervised learning methods, while traditional approaches like static and dynamic analysis encounter scalability and feature engineering challenges. The study evaluates seven classification algorithms on a dataset comprising 29,999 instances and 180 features. XGBoost emerges as the top-performing algorithm with 80.03% accuracy. However, limitations such as imbalanced data, missing values, and computational constraints for specific algorithms are acknowledged. The study highlights the efficiency of machine learning in identifying Android malware based on requested permissions but suggests further research to address these limitations for more robust detection systems.

In a study by Rana et al. [8], the following tree-based algorithms, Random Forest, Decision Tree, Gradient Tree Boosting and Extremely Randomized Tree, were assessed to detect Android Malware. Selecting features based on substrings is mainly used for creating classifiers. The researcher used three distinct substrings for each feature: one word, two words, and three words, respectively. The researcher used the DREBIN dataset, which comprised 11,120 apps with data and metadata, equally split between malicious and benign apps. The researcher assessed the classifiers' performance using precision, recall, accuracy, and F1 score criteria. The Random Forest algorithm achieved a competitive accuracy rate of 97.24% and performed well in other metrics. Therefore, it is the best algorithm for all substring situations. Blending other terms in the last words of different app features adversely affected the performance of the classifiers. Hence, optimizing the app's performance requires careful attention to this aspect.

McLaughlin et al. [25], researchers employed convolution neural networks to process an Android app's raw Dalvik byte code to detect malware on the device. They trained their neural network using the Torch environment and optimized it with RMSProp. According to their claims, one of the main advantages of their approach is that it eliminates the need for manually designed malware features.

A group of developers conducted a study on a newly developed application classification technique called DroidCat [26]. The study found that app execution traces can provide valuable information on various features, although not all may be useful. The developers defined 122 metrics by characterizing application behaviours using these execution traces. The creators of DroidCat designed it to overcome challenges that most existing peer approaches face. The results demonstrated that DroidCat outperformed other approaches

in detecting malware and categorizing families.

The study found that malware invokes SDK APIs more frequently than benign Android apps from user code or third-party libraries. It also discovered that malicious apps tend to log less than benign ones. The results showed that DroidCat outperformed state-of-the-art malware detection and categorization techniques, achieving a mean F1 accuracy of 97.39%. Researchers noted that DroidCat achieved an impressive precision rate of 96.83% for malicious apps. DroidCat has a high chance of detecting any unknown app with an accuracy rate of 96.83%. A reliable antivirus tool in Virus Total would also detect the same app.

Wang et al. (2019) proposed a hybrid model aimed at improving the accuracy and efficiency of Android malware detection on a large scale. The proposed model relied on two crucial components: deep autoencoder (DAE) and convolution neural network (CNN). Virus Total confirmed that all 10,000 apps crawled by the Anzhi Play Store were benign. Virus Share collected 13,000 malicious apps, resulting in a dataset of 23,000 apps used to train and test various models. The researchers reconstructed the high-dimensional features of the apps to boost the accuracy of Android malware detection, reducing the number of features from 34,570 to 413. The researchers deployed a serial convolution network architecture (CNN-S) to detect malware in apps, which achieved the highest accuracy of 99.82% compared to traditional methods and other CNN architectures. DAE was combined with CNN to create the DAE-CNN model to achieve the same performance while reducing the training time. The results showed that training time was reduced by 83% compared to CNN-S, resulting in improved accuracy and efficiency.

Jerome et al. [7] examined machine learning-driven device malware detection. They introduced a tool that autonomously extracts characteristics from Android smartphones and analyzed emulator-centric and device-centric detection methodologies across trials. Their Dataset comprised 2444 applications, with half sourced from the Android Malware Genome Project and the other half sourced from Intel Security. By employing InfoGain, the researchers identified 100 features out of a total of 178 features. They focused solely on these selected features. During their experimentation, the team grouped 100 features into five categories: top 20, 40, 60, 80, and top 100. They analyzed these features using several classifiers, such as MLP, SVM, Random Forest, Naive Bayes, PART, J48 decision tree model, and Simple Logistics. Subsequently, they evaluated the applications in emulator—and device-based settings to assess and compare their performance levels. The findings indicated that the device-based approach is notably more effective in analyzing a percentage of apps than the emulator environment.

Table 4.1 compares the related work’s primary Methodology, Contributions and limitations. The earlier studies demonstrate that the Dataset needed to be revised and unjustifiable given the variety of apps available in the Google Play Store.

4.4 Methodology

This study presents a new method for detecting malware using Machine Learning (ML) algorithms and ensemble prediction techniques. The study aims to identify the most effective approach by evaluating various performance metrics through comprehensive experimentation. Figure 4.1 depicts the overall Methodology. This approach provides a novel and promising solution to the challenging issue of malware detection.

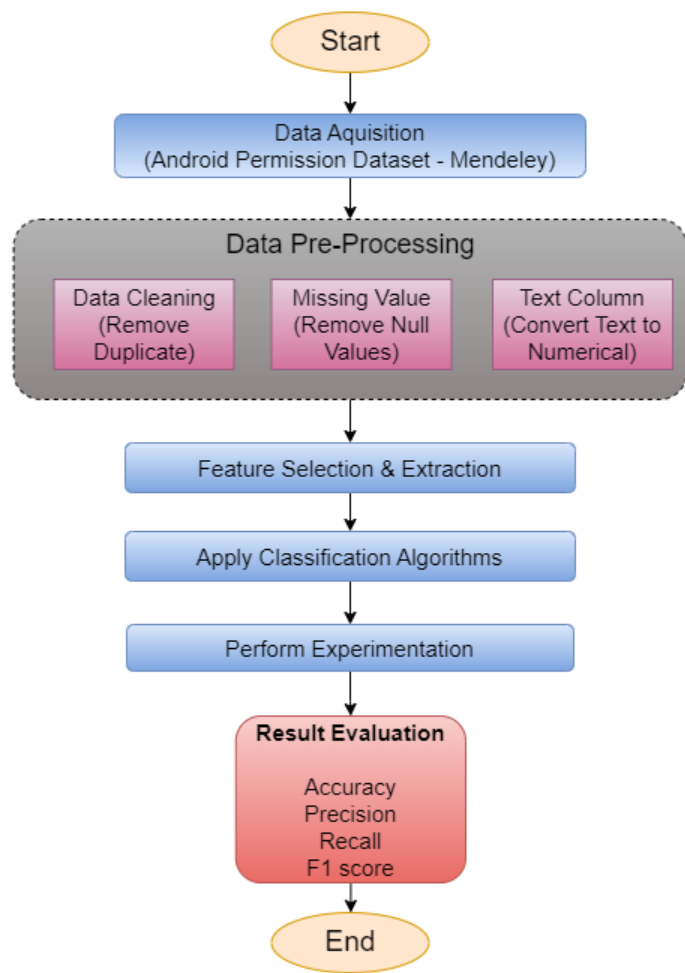


Figure 4.1: Flow Diagram

Table 4.1: Related Work

Research Work	Methodology	Contribution	Limitation
Rawat et al. [15]	Machine learning algorithms focusing on permissions requested by apps	XGBoost emerged as the top-performing algorithm with 80.03% accuracy.	Imbalanced data, missing values, and computational constraints for specific algorithms are acknowledged.
Rana et al. [8]	Tree-based algorithms and substring-based feature selection	Random Forest algorithm achieved 97.24% accuracy on 11,120 apps in DREBIN Dataset.	Static analysis focus, outdated Dataset, and exclusive tree classifiers evaluation.
McLaughlin et al. [25]	CNN on raw Dalvik byte code	Eliminate manual malware feature identification with the 2163 Applications Dataset.	This method may not generalize well to new, unseen malware variants.
DroidCat [26]	App classification using execution traces	Used four datasets with approximately 5000 applications and achieved 97.39% F1 accuracy.	Imbalanced training datasets may affect the generalizability of results.
Wang et al. [27]	Hybrid model with DAE and CNN	Achieved 99.82% accuracy in malware detection.	lack of real-world deployment and evaluation.
Jerome et al. [7]	Device-based detection with dynamic analysis	2,444 Android applications were examined for malware using opcode sequences.	lack of evaluation on diverse datasets and real-time detection scenarios.

4.4.1 Dataset

The Dataset consists of comma-separated values of 29999 instances, each with 180 features [14]. Among these features, one output variable classifies the Android package as benign or malicious. The package has two categories of permissions: safe and dangerous. Most of its features are related to these permissions. Additionally, the Dataset includes features that provide general information about the packages, such as "app", "description", "related apps", "price", and others.

The Dataset contains a mix of string and integer values. One of the columns in the Dataset, called "Related Apps", has 755 null values. Additionally, the Dataset's dependent variable indicates an imbalance ratio of 2:1 for malicious and benign apps. Dataset distribution is shown in Figure 4.2, where "Red" represents malicious Android packages and "Green" represents benign Android packages.

4.4.2 Pre-processing

This paper performed standard exploratory data analysis, examining column data types, generating a numerical statistical summary, and searching for empty values. It is simple

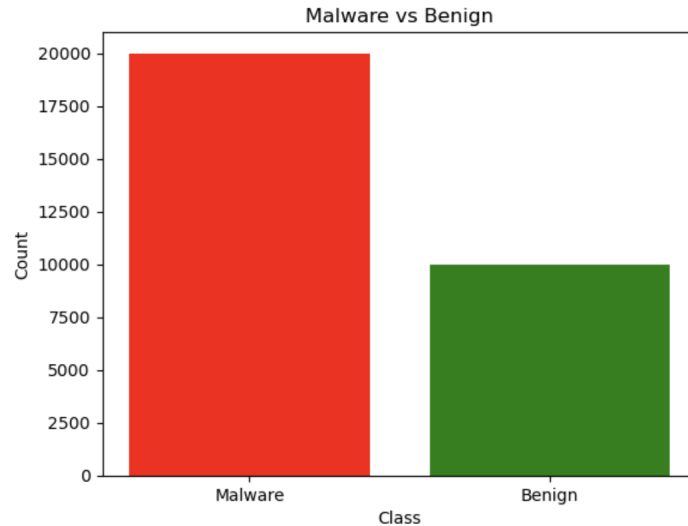


Figure 4.2: Dataset Distribution

to rearrange the "app," "description," and "category" columns to trick consumers. As a result, this work refrained from using such data for analysis. The Dataset has 204 empty entries in the "Dangerous permissions count" column and 755 empty values in the "Related apps" column. For convenience, this work replaced the value in related apps. This work decided not to use generic value replacement techniques for "dangerous permissions" because they might cause harm.

1. *Handling Missing Values:* Adding values to the Dataset can help the effectiveness of analysis and modelling efforts. It's essential to handle them appropriately to ensure the integrity of the data. This work identifies and removes rows with missing values in the preprocessing pipeline using pandas' `dropna()` function. Removing rows with missing values ensures that the Dataset used for analysis and modelling is complete and does not introduce biases or errors.
2. *Feature Selection:* Feature selection is a crucial step in preprocessing, as it involves identifying the most relevant features for the analysis and model training. This approach prioritizes features based on their occurrence frequency and relevance to identifying scam applications. The system drops unnecessary columns to streamline the Dataset. Removing irrelevant columns reduces the Dataset's dimensionality and focuses on features likely to impact the task significantly.
3. *Text Data Transformation:* Various techniques transform textual columns, such as app names, descriptions, and package names, into numerical features for analysis and modelling. These techniques include counting uppercase letters and periods and analyzing word patterns. Machine learning algorithms can effectively process and analyze textual information by transforming text data into numerical features.
4. *Encoding Categorical Variables:* Categorical variables, such as app categories, must be encoded numerically to ensure compatibility with machine learning algorithms. Label encoding converts categorical variables to numerical representations for better analysis by machine learning models.
5. *Feature Engineering:* Feature engineering creates or modifies features to enhance a dataset's predictive power. In this approach, additional features are generated

based on patterns and insights obtained from the data.

For instance, detecting the word "free" in app names can create new features that provide additional information, enhancing machine learning models' performance.

6. *Data Visualization*: Data visualization is crucial in exploring and understanding datasets. Various visualization techniques, including count plots, heatmaps, scatter plots, and box plots, are utilized to gain insights into data distributions, correlations, and outliers. For example, the graph illustrates the relationship between application price and category in Figure 4.3.

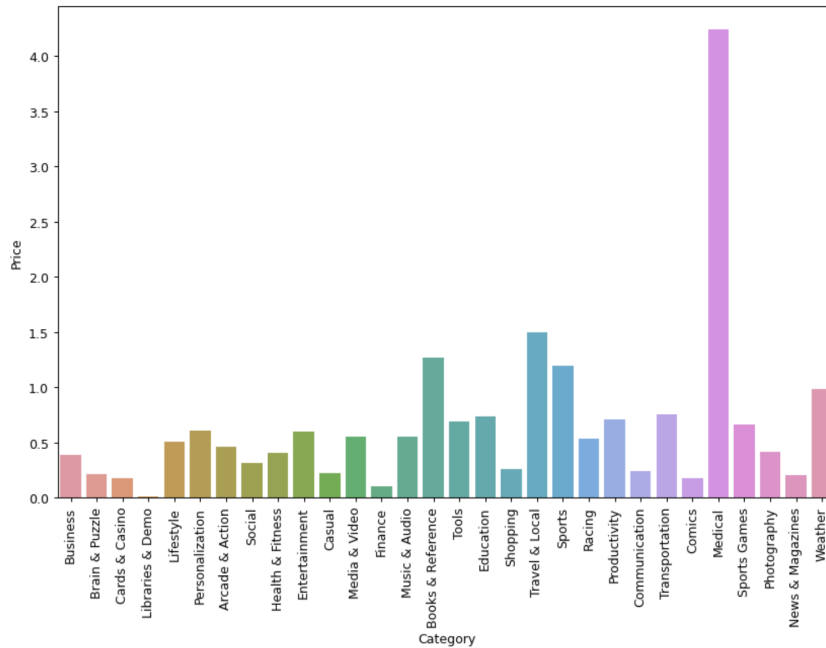


Figure 4.3: Price vs Category of applications

This paper analyzes the Dataset using six different classification algorithms. These algorithms include a Gradient boosting classifier, Ridge classifier, random forest classifier, MLP classifier, Bagging classifier with decision tree, and ADA boost classifier. Whenever appropriate, hyperparameter tuning is applied to improve the algorithms' performance. The Randomized Search object applied a default 5-fold cross-validation. The work split the original Dataset into 80-20 training and testing sets to evaluate algorithm performance.

4.4.3 Hyperparameter Tuning

After preprocessing, the system runs each algorithm using the default hyperparameters. The default hyperparameters are then analyzed, and based on the findings, the hyperparameters are adjusted to enhance the outcomes.

4.4.4 Experimentation

The Experimentation Phase is an in-depth analysis of the performance of several additional machine learning algorithms employed in the classification task. Let's discuss each algorithm's accuracy, precision, recall, and F1 score, which will illuminate its strengths

and weaknesses. Experimenting with different machine learning algorithms to obtain an accurate model for prediction is illustrated in Figure 4.4.

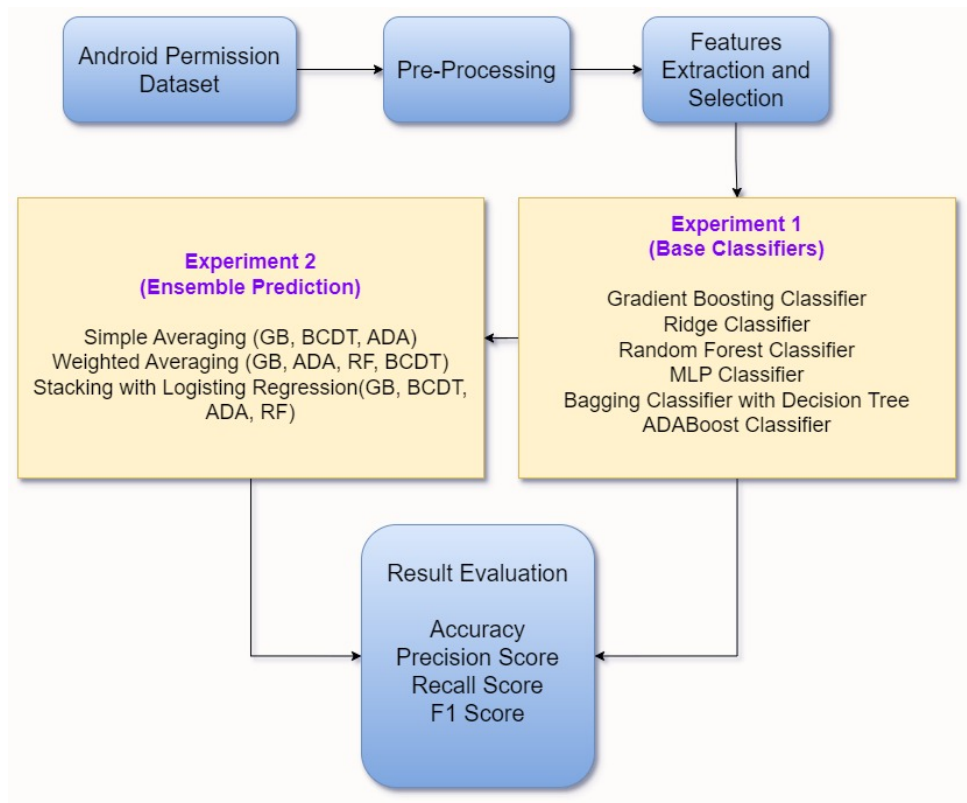


Figure 4.4: Different Experimental Phases

Experiment 1: Base Classifiers

- a. **Gradient Boosting Classifier (GB):** Gradient Boosting Classifier, a sophisticated ensemble learning technique, showcased exceptional performance in the classification task. It builds a robust predictive model by combining decision trees iteratively. The algorithm demonstrated a notable accuracy of 82.01%, surpassing several other methods in the study. Its precision score is 0.8225, recall is 0.8201, and F1 score is 0.8211, underscoring its effectiveness in accurately identifying positive and negative instances.
- b. **Ridge Classifier:** Despite being a linear model, the Ridge Classifier exhibited decent performance with an accuracy of 71.15%. This algorithm applies L2 regularization to mitigate the effects of multicollinearity in the feature space. It performed well overall, with a precision score of 0.6998, a recall score of 0.7115, and an F1 score of 0.7019. However, compared to more flexible algorithms, it needed to be improved in capturing complex nonlinear relationships within the data.
- c. **Random Forest Classifier (RF):** Random Forest Classifier emerged as a strong performer with an accuracy of 80.53%. By aggregating the predictions of multiple decision trees trained on random subsets of the data [27], it effectively mitigates overfitting and yields robust predictions. Its precision score of 0.8062, recall score of 0.8053, and F1 score of 0.8057 highlight its ability to balance precision and recall while maintaining high accuracy.

- d. **MLP Classifier:** The MLP (Multi-Layer Perceptron) Classifier, a type of artificial neural network [28], delivered a competitive performance with an accuracy of 76.81%. Its architecture, comprising multiple layers of interconnected neurons, enables it to learn complex patterns in the data. Despite its lower accuracy than some ensemble methods, its precision score of 0.7889, recall score of 0.7681, and F1 score of 0.7731 demonstrate its effectiveness in capturing intricate relationships within the feature space.
- e. **Bagging Classifier with Decision Tree (BCDT):** Using a Bagging Classifier combined with Decision Trees produced encouraging outcomes, reaching an accuracy rate of 81.24%. Bagging, short for Bootstrap Aggregating, includes training base learners on bootstrapped training datasets and combining their forecasts. With a precision rate of 0.8143, a recall rate of 0.8124 and an F1 score of 0.8132, this ensemble technique showcased performance in managing noise and overfitting issues.
- f. **ADABOOST Classifier (ADA):** The ADABOOST Classifier demonstrated strong performance, achieving an accuracy of 80.27%. AdaBoost, short for Adaptive Boosting, is an iterative ensemble method that adjusts the weights of misclassified instances in subsequent iterations to focus on difficult-to-classify samples. Its precision score is 0.802, recall score is 0.8027, and an F1 score of 0.8023 highlight its ability to effectively classify both positive and negative instances.

Experiment 2: Ensemble Prediction

- a. **Ensemble Predictions using Simple Averaging (GB, BCDT, ADA):** Combining the predictions of classifiers like Gradient Boosting, Bagging, and AdaBoost and applying a simple averaging method after ensemble models improves results, reaching an accuracy level of 82.16%. This ensemble approach utilizes each model's abilities, resulting in a precision score of 0.8742, a recall score of 0.8545, and an F1 score of 0.8643. It excels in categorizing instances across classes.
- b. **Ensemble Predictions using Weighted Averaging (GB, ADA, RF, BCDT) (GARF):** By incorporating Random Forest through weighted averaging, the ensemble predictions achieved an impressive accuracy of 82.38%. This comprehensive ensemble method leveraged the strengths of each algorithm, including Gradient Boosting, Bagging, and AdaBoost classifiers along with Random Forest. The ensemble method outperformed the others with a precision score of 0.8762, a recall score of 0.8545, and an F1 score of 0.8653. Additionally, it accurately classified instances across multiple classes while maintaining high precision.

- c. **Stacking with Logistic Regression:**

Using Logistic Regression in a stacked approach achieved an accuracy of 81.86%. This technique combines the abilities of base models (GB, BCDT, ADA, RF) by training a meta-model-like Logistic Regression on their predictions. The precision score, recall, and F1 scores are 0.8188, 0.8186, and 0.8187, respectively, indicating a balance between precision and recall for classifying instances across different classes while maintaining high precision levels.

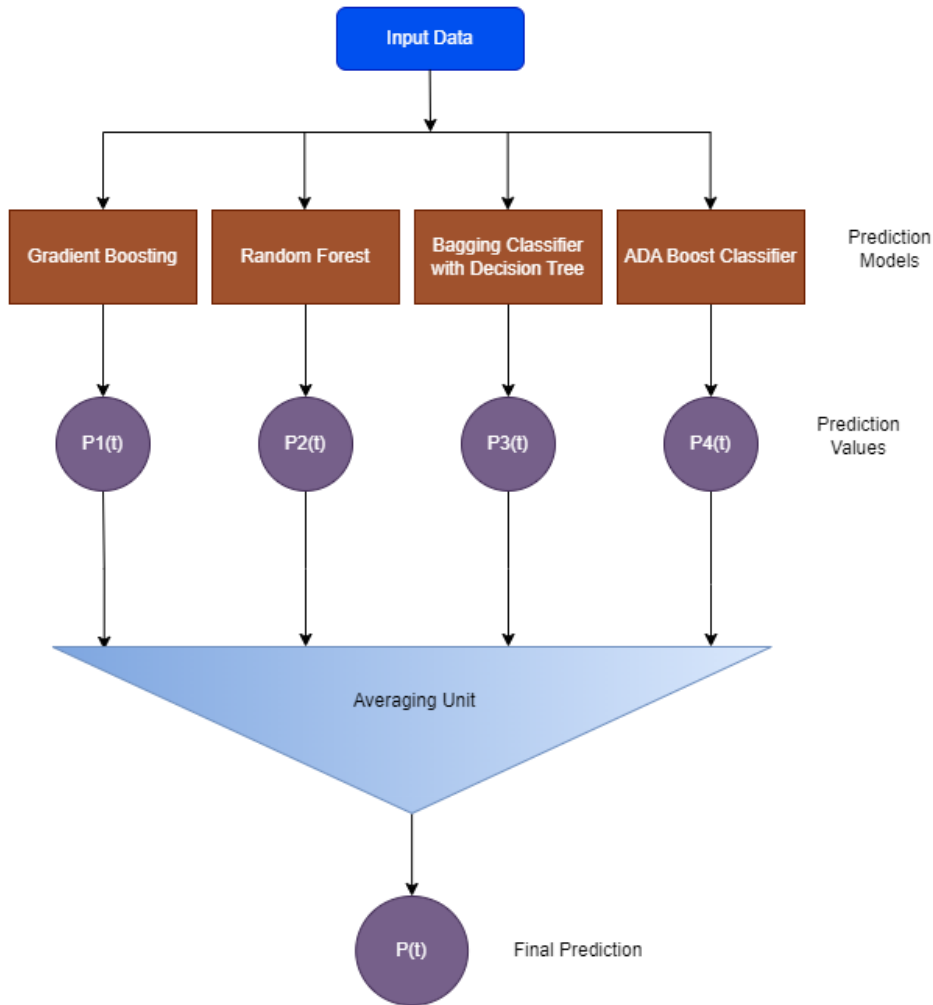


Figure 4.5: GARB (*Gradient Boosting classifier, AdaBoost, Random Forest and Bagging classifier with a decision tree*) Model

4.5 Proposal of GARB Model

The GARB Model is a method used for classification tasks that combines base models, like Gradient Boosting, ADA Boost Random Forest and Bagging Classifiers with a Decision Tree. Its goal is to utilize a variety of algorithms to improve classification accuracy and reliability. One interesting feature of the GARB model is its stacking with averaging technique. This approach combines predictions of base models to produce an overall prediction, as shown in Figure 4.5. Assigning weights based on individual model performance reduces biases.

In the context of stacking with averaging, stacked generalization involves combining estimator predictions and inputting them into an estimator trained using cross-validation. This approach helps minimize estimator biases and harnesses algorithms' strengths for reliable predictions.

The Stacking Classifier and Stacking Regression frameworks offer tools for professionals seeking to enhance accuracy in classification and regression tasks by combining models. The proposed GARB Model, which uses stacking with averaging, offers an ensemble learning framework designed explicitly for classification tasks. By combining the insights of multiple models and using stacking techniques, the GARB Model has the potential to achieve higher accuracy in classifying and handling challenges in practical scenarios.

Chapter 5

RESULTS and DISCUSSION

5.1 Android App Permission Detector based on Machine Learning Models

The results of this study demonstrate the successful integration of the App Permission Classification with Efficient Clustering (APEC) Model [1] into an Android application for permission classification, allowing users to distinguish between safe and unsafe app permissions. Notably, the model achieved an impressive accuracy rate of 93.8% using Decision Tree and 95.8% using Random Forest classifiers. The DBSCAN clustering algorithm, an unsupervised learning methodology, effectively rated apps within various categories. The classification model, trained on 100% of data from a dataset comprising two million apps and tested with a real-time Google Play Store application, empowers users to manage permissions, safeguarding their privacy.

Table 5.1: Application Testing with Proposed Application

Device Android Version	Category	App Name	Unsafe Permissions	Unsafe Permissions that Users Can Enable/Disable
LG / 12	Books and Ref- erence	Freed Audio- books	WAKE_LOCK FOREGROUND_SERVICE WRITE_EXTERNAL_STORAGE ACCESS_COARSE_LOCATION ACCESS_FINE_LOCATION	WRITE_EXTERNAL_STORAGE ACCESS_COARSE_LOCATION ACCESS_FINE_LOCATION
Samsung / 10	Beauty	TST- Malaysia	CHANGE_NETWORK_STATE READ_CONTACTS WRITE_EXTERNAL_STORAGE CALL_PHONE WAKE_LOCK GET_TASKS	READ_CONTACTS WRITE_EXTERNAL_STORAGE CALL_PHONE
POCO / 13	Board	Fun101 Okey	WRITE_EXTERNAL_STORAGE RECORD_AUDIO MODIFY_AUDIO_SETTINGS WAKE_LOCK KILL_BACKGROUND_PROCESS FOREGROUND_SERVICE	WRITE_EXTERNAL_STORAGE RECORD_AUDIO

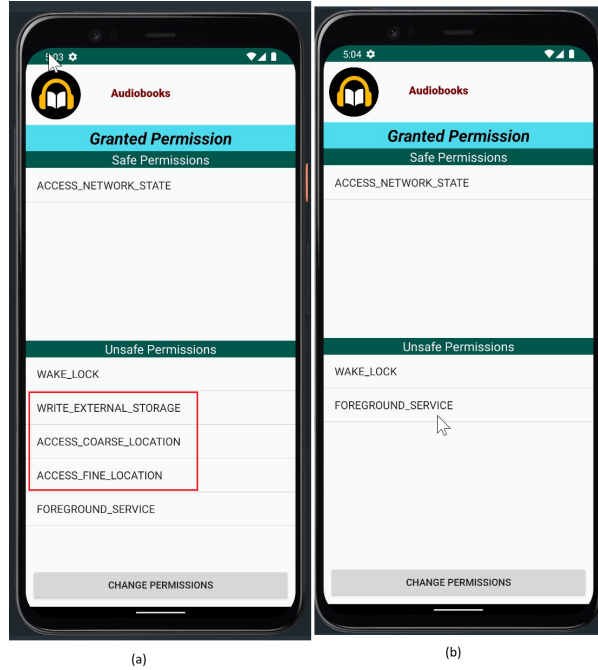


Figure 5.1: The interface of Permission Detector. In Fig. (a), Audiobooks show granted unsafe permission, of which three are user-authorized. In Fig. (b), After turning off User-authorized permission from setting and refresh the app.

Testing on Google Play Store applications across all 48 different categories [29] revealed the accuracy for determining safe and unsafe permissions, enabling users to make informed decisions regarding permission management. Furthermore, rigorous testing on various models of Android phones, including the LG G8X, Motorola, Realme, Redmi, OnePlus 6, POCO, OnePlus, and Samsung, demonstrated the versatility and robustness of the system. These results underscore the system’s effectiveness across various Android devices, enhancing its applicability in real-world scenarios. For example, as depicted in Fig. 5.1, the Freed Audiobooks application determines the granted unsafe permissions for which the user authorizes only a few. After turning off user-authorized permissions, the Permission Detector app removes those unsafe permissions. This paper displays testing results in Table 5.1, and you can locate the complete results in [30]. These results underscore the potential of the APEC Model [1] in bolstering mobile app security and user privacy while fostering transparency between users and developers.

5.2 Android App Malware Detection using Stacking of Machine Learning Algorithms

The performance of various classification algorithms, including Gradient Boosting Classifier, Ridge Classifier, Random Forest Classifier, MLP Classifier, Bagging Classifier with Decision Tree, AdaBoost Classifier, Ensemble Predictions using simple averaging (GB, BG, ADA), Ensemble Predictions using Weighted Averaging (GB, ADA, RF, BG) (referred to as the GARB Model), and Stacking with Logistic Regression, is evaluated and compared using metrics such as accuracy, precision score, recall score, and F1 score. The results are summarized in Table 5.2.

Table 5.2: Performance Comparison of Different Algorithms

Algorithms	Accuracy	Precision Score	Recall Score	F1 Score
Rawat et al. (XGBoost Classifier) [15]	0.8003	0.8477	0.8532	0.8505
Rawat et al. (CatBoost Classifier) [15]	0.7958	0.8378	0.8622	0.8499
Gradient Boosting Classifier	0.8201	0.8225	0.8201	0.8211
Ridge Classifier	0.7115	0.6998	0.7115	0.7019
Random Forest Classifier	0.8053	0.8062	0.8053	0.8057
MLP Classifier	0.7681	0.7889	0.7681	0.7731
Bagging Classifier with Decision Tree	0.8124	0.8143	0.8124	0.8132
AdaBoost Classifier	0.8027	0.802	0.8027	0.8023
Ensemble Predictions using simple averaging (GB, BG, ADA)	0.8216	0.8742	0.8545	0.8643
Stacking with Logistic Regression	0.8187	0.8188	0.8186	0.8187
Proposed GARB Model	0.8238	0.8762	0.8545	0.8653

The GARB Model, which integrates Gradient Boosting, ADA Boost, Random Forest, and Bagging classifier with Decision Tree, along with stacking using weighted averaging, yielded the highest accuracy among all models, achieving an accuracy of **82.38%**. This ensemble method capitalizes on the diverse strengths of each algorithm and effectively combines their predictive capabilities to enhance classification performance.

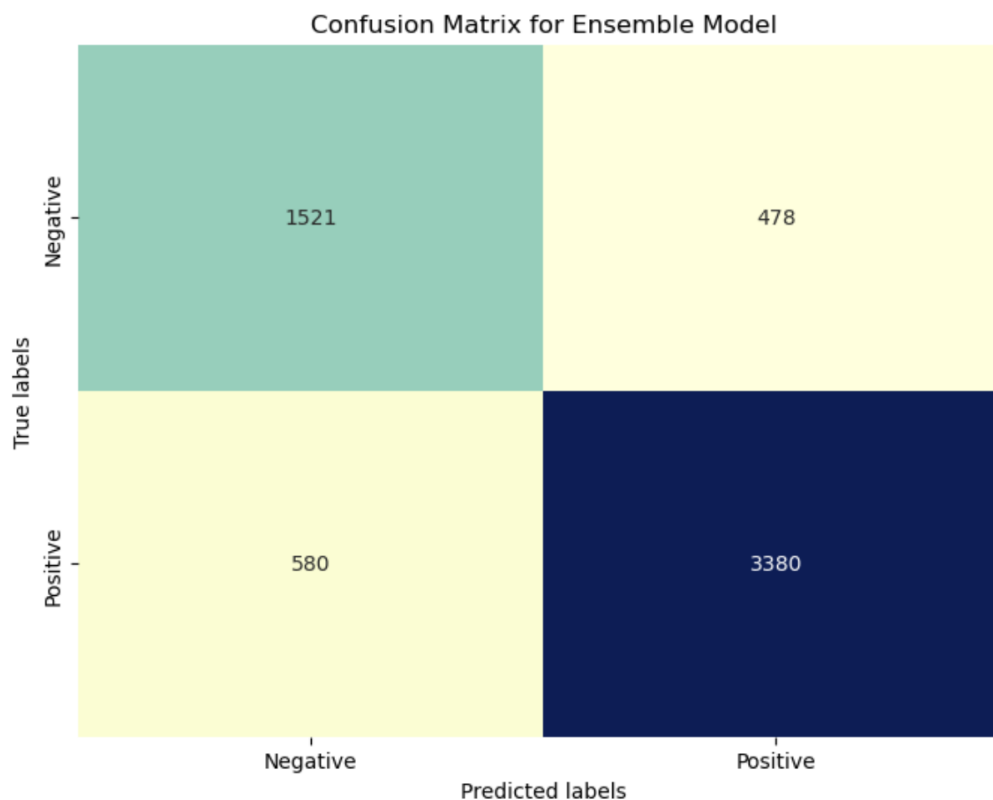


Figure 5.2: Confusion Matrix of GARB Model

In addition, the GARB Model showed precision, recall, and F1 scores of 0.8762, 0.8545, and 0.8653, respectively. GARB demonstrates its ability to classify instances across categories while maintaining precision accurately. Figure 5.2 presents the confusion matrix of the GARB Model, which shows the supplementary analysis.

Stacking with Logistic Regression, in particular, has shown promising results with an accuracy rate of **81.87%**. This approach utilizes stacked generalization to merge predictions from estimators and fine-tune an estimator through cross-validation. With precision, recall, and F1 scores of 0.8188, 0.8186, and 0.8187, respectively, Stacking with Logistic Regression balances precision and recall, showcasing its effectiveness in categorizing instances while reducing biases.

Overall, the experimental findings highlight the effectiveness of learning techniques such as the GARB Model and Stacking with Logistic Regression in enhancing classification performance across various domains. These results emphasize the importance of employing advanced algorithms and ensemble methods to achieve predictive accuracy and resilience in practical scenarios.

For a more detailed overview of the results and discussions regarding algorithm performance, refer to Table 5.2, which provides insights into their metrics and comparative analyses of previous related work and current work.

Chapter 6

CONCLUSION AND FUTURE SCOPE

The evolution of the Android operating system has witnessed considerable advancements in app permissions and security mechanisms. Despite these strides, the persistent menace of fraudulent applications infiltrating the Google Play Store raises pertinent concerns regarding the overall integrity of the Android ecosystem. This research underscores the pressing need for innovative solutions to augment user control and fortify mobile security.

The introduction of the Android App Permission Detector, underpinned by the APEC model, represents a proactive response to address these security challenges. By equipping users with a robust toolset for evaluating and managing app permissions, this solution not only empowers individuals to make informed decisions about their digital privacy but also serves as a bulwark against potential security breaches arising from unauthorized access to sensitive data. Thus, the Android App Permission Detector emerges as a beacon of hope in the ongoing battle to safeguard the integrity of the mobile app landscape.

Moreover, the efficacy demonstrated by advanced classification algorithms such as the GARB Model and Stacking with Logistic Regression in identifying and categorizing malware Android Packages underscores the pivotal role of cutting-edge technology in bolstering mobile security measures. By leveraging sophisticated machine learning techniques, these models can discern subtle patterns indicative of malicious intent, enabling preemptive action to mitigate security risks effectively.

Looking ahead, it is imperative to sustain momentum in research and development efforts aimed at refining permission classification methodologies and exploring diverse ensemble learning techniques. Such endeavours promise to foster a more secure and privacy-aware mobile app ecosystem wherein users can confidently engage with digital applications without compromising their personal data or system integrity. Thus, by embracing innovation and collaboration, stakeholders within the Android community can collectively strive towards a future characterized by enhanced mobile security and user empowerment.

Bibliography

- [1] P. S. Rawal and D. Sethia, “APEC: App Permission classification with Efficient Clustering,” 2023, international Conference On Computational Intelligence For Information, Security And Communication Applications (CIISCA - 2023) (Accepted in conference).
- [2] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, “A review of android malware detection approaches based on machine learning,” *IEEE Access*, vol. 8, pp. 124 579–124 607, 2020.
- [3] A. Developers, “Permission on android.” [Online]. Available: <https://developer.android.com/guide/topics/permissions/overview>
- [4] A. S. Nabila Popal, “IDC - smartphone market share - market share,” <https://www.idc.com/promo/smartphone-market-share>, 08 2023, [Last accessed on September 15, 2023].
- [5] IANS, “Android now powers over 3 billion devices worldwide,” https://bit.ly/android_devices_sundar_pichai, 07 2022, [Last accessed on October 07, 2023].
- [6] X. Su *et al.*, “An informative and comprehensive behavioral characteristics analysis methodology of android application for data security in brain-machine interfacing,” <https://scite.ai/reports/10.1155/2020/3658795>, 03 2020, [Last accessed on October 10, 2023].
- [7] Jerome *et al.*, “Using opcode-sequences to detect malicious android applications,” in *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 914–919.
- [8] M. S. Rana, S. S. M. M. Rahman, and A. H. Sung, “Evaluation of tree based machine learning classifiers for android malware detection,” in *Computational Collective Intelligence: 10th International Conference, ICCCI 2018, Bristol, UK, September 5-7, 2018, Proceedings, Part II 10*. Springer, 2018, pp. 377–385.
- [9] K. Olmstead and M. Atkinson, “An analysis of android app permissions. pew research center: Internet, science & tech.” https://bit.ly/pewresearch_android, 11 2015, [Last accessed on October 12, 2023].
- [10] Yang *et al.*, “Pradroid: Privacy risk assessment for android applications,” in *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, 2021, pp. 90–95.
- [11] “Google, Android 12 Features Overview,” <https://bit.ly/3QK9q3t>, 10 2021, [Last accessed on October 1, 2023].

- [12] “Google, Android 13 Features Overview,” <https://bit.ly/40pNqOz>, 08 2022, [Last accessed on October 18, 2023].
- [13] Z. Wang, Q. Liu, and Y. Chi, “Review of android malware detection based on deep learning,” *IEEE Access*, vol. 8, pp. 181 102–181 126, 2020.
- [14] M. Arvind, “Android permission dataset,” *Mendeley Data*, vol. 1, 2018.
- [15] S. Rawat, R. Phira, and P. Natu, “Use of machine learning algorithms for android app malware detection,” in *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEEC-COT)*, 2021, pp. 448–454.
- [16] Manzil *et al.*, “Covid-themed android malware analysis and detection framework based on permissions,” in *2022 International Conference for Advancement in Technology (ICONAT)*, 2022, pp. 1–5.
- [17] S. Arshad *et al.*, “Samadroid: A novel 3-level hybrid malware detection model for android operating system,” *IEEE Access*, vol. 6, pp. 4321–4339, 2018.
- [18] L. Sun *et al.*, “Sigpid: significant permission identification for android malware detection,” in *2016 11th International Conference on Malicious and Unwanted Software (MALWARE)*, 2016, pp. 1–8.
- [19] Z. Wang, Li *et al.*, “Multilevel permission extraction in android applications for malware detection,” in *2019 International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2019, pp. 1–5.
- [20] Samah *et al.*, “On the effectiveness of application permissions for android ransomware detection,” in *2020 6th Conference on Data Science and Machine Learning Applications (CDMA)*, 2020, pp. 94–99.
- [21] “Permission Manager created by NorthRiver,” <https://play.google.com/store/apps/details?id=com.agooday.permission>, 2019.
- [22] “Permission Pilot created by darken,” <https://play.google.com/store/apps/details?id=eu.darken.myperm>, 2022.
- [23] “App Permission Manager created by Micro Inc,” <https://play.google.com/store/apps/details?id=com.assistant.android.permission.manager>, 2021.
- [24] A. Felt *et al.*, “Android permissions: User attention, comprehension, and behavior,” *SOUPS 2012 - Proceedings of the 8th Symposium on Usable Privacy and Security*, 07 2012.
- [25] McLaughlin *et al.*, “Deep android malware detection,” in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, ser. CODASPY ’17, 2017, p. 301–308. [Online]. Available: <https://doi.org/10.1145/3029806.3029823>
- [26] H. Cai, N. Meng, B. Ryder, and D. Yao, “Droidcat: Effective android malware detection and categorization via app-level profiling,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 6, pp. 1455–1470, 2019.
- [27] W. Wang, M. Zhao, and J. Wang, “Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network,”

Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 8, pp. 3035–3043, 2019. [Online]. Available: <https://doi.org/10.1007/s12652-018-0803-6>

- [28] P. D. Nandanwar and D. S. B. Dhonde, “A novel approach to cervical cancer detection using hybrid stacked ensemble models and feature selection,” *International Journal of Electrical Engineering and Robotics*, vol. 11, no. 2, pp. 582–589, 2023.
- [29] Google, “Android category and tags for your app or game,” https://bit.ly/android_category, [Last accessed on October 05, 2023].
- [30] “Android Permission Detector Testing Results,” https://bit.ly/testing_result.



DELHI TECHNOLOGICAL UNIVERSITY
 (Formerly Delhi College of Engineering)
 Shahbad Daultapur, Main Bawana Road, Delhi-42

PLAGIARISM VERIFICATION

Title of the Thesis Enhancing Android Security: Machine Learning Approaches for App Permissions and Malware Detection
 Total Pages 46 Name of the Scholar Jaikishan Mahanty

Supervisor (s)

- (1) Dr. Divyashikha Sethia
 (2) _____
 (3) _____

Department Software Engineering

This is to report that the above thesis was scanned for similarity detection. Process and outcome is given below:

Software used: Turnitin Similarity Index: 11%, Total Word Count: 12007

Date: 24/05/24

[Signature]
 Candidate's Signature

[Signature]

Signature of Supervisor(s)

PAPER NAME

Jaikishan_Thesis_10May24_PlagRep

WORD COUNT

12007 Words

CHARACTER COUNT

71529 Characters

PAGE COUNT

46 Pages

FILE SIZE

2.2MB

SUBMISSION DATE

May 10, 2024 5:05 PM GMT+5:30

REPORT DATE

May 10, 2024 5:06 PM GMT+5:30

● 11% Overall Similarity*D. W. Seth*

The combined total of all matches, including overlapping sources, for each database.

- 7% Internet database
- 4% Publications database
- Crossref database
- Crossref Posted Content database
- 7% Submitted Works database

● Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less than 8 words)
- Manually excluded text blocks

● 11% Overall Similarity

Top sources found in the following databases:

- 7% Internet database
- 4% Publications database
- Crossref database
- Crossref Posted Content database
- 7% Submitted Works database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	developer.android.com Internet	2%
2	researchgate.net Internet	<1%
3	Shaurya Rawat, Rushang Phira, Prachi Natu. "Use of Machine Learning ... Crossref	<1%
4	Praveen Singh Rawal, Divyashikha Sethia. "App Permission Classificati... Crossref	<1%
5	Napier University on 2022-05-12 Submitted works	<1%
6	doctorpenguin.com Internet	<1%
7	fedetd.mis.nsysu.edu.tw Internet	<1%
8	hindawi.com Internet	<1%

9	University of Hertfordshire on 2024-04-29 Submitted works	<1%
10	Athlone Institute of Technology on 2021-04-04 Submitted works	<1%
11	University of Central Lancashire on 2021-01-11 Submitted works	<1%
12	"Recent Trends in Image Processing and Pattern Recognition", Springe... Crossref	<1%
13	iieta.org Internet	<1%
14	University of Southern Mississippi on 2021-07-02 Submitted works	<1%
15	link.springer.com Internet	<1%
16	search.bvsalud.org Internet	<1%
17	stax.strath.ac.uk Internet	<1%
18	dspace.dtu.ac.in:8080 Internet	<1%
19	Rena Lavranou, Stylianos Karagiannis, Aggeliki Tsohou, Emmanouil Ma... Crossref	<1%
20	University of New South Wales on 2022-12-04 Submitted works	<1%

- 21 **Kaijun Liu, Shengwei Xu, Guoai Xu, Miao Zhang, Dawei Sun, Haifeng Liu...** <1%
Crossref

- 22 **University of Huddersfield on 2021-09-27** <1%
Submitted works

- 23 **Uttar Pradesh Technical University on 2019-06-15** <1%
Submitted works

- 24 **Visvesvaraya National Institute of Technology on 2021-07-06** <1%
Submitted works

- 25 **University of Hull on 2023-03-20** <1%
Submitted works

- 26 **Indiana University on 2023-07-13** <1%
Submitted works

- 27 **College of Engineering Trivandrum on 2019-11-21** <1%
Submitted works

- 28 **Natalia Baxevanou, Sotiria Triantafyllia Sotirhou, Konstantinos Limnioti...** <1%
Crossref

- 29 **SHAPE (VTC college) on 2019-06-04** <1%
Submitted works

- 30 **University College London on 2017-09-03** <1%
Submitted works

- 31 **University of Bedfordshire on 2023-07-28** <1%
Submitted works

- 32 **khazna.ku.ac.ae** <1%
Internet

33	escholar.manchester.ac.uk Internet	<1%
34	mdpi.com Internet	<1%
35	"Data Engineering and Communication Technology", Springer Science ... Crossref	<1%
36	Chester College of Higher Education on 2023-10-05 Submitted works	<1%
37	Chester College of Higher Education on 2024-02-08 Submitted works	<1%
38	Heriot-Watt University on 2019-04-03 Submitted works	<1%
39	Huasong Meng, Vrizlynn L.L. Thing, Yao Cheng, Zhongmin Dai, Li Zhan... Crossref	<1%
40	Saba Arshad, Munam Ali Shah, Ayesha Siddiq, Hafsa Maryam, Abdul ... Crossref	<1%
41	University of Hertfordshire on 2023-04-23 Submitted works	<1%
42	University of Westminster on 2023-09-07 Submitted works	<1%
43	openknowledge.worldbank.org Internet	<1%
44	srmengineeringcollege on 2024-03-28 Submitted works	<1%

DECLARATION

I hereby certify that the work which is presented in Major Project-II entitled **Enhancing Android Security: Machine Learning Approaches for App Permissions and Malware Detection** in fulfillment of the requirement for the award of the Degree of Master of Technology in Software Engineering and submitted to the Department of Software Engineering, Delhi Technological University, Delhi is an authentic record of my own, carried out during a period from January to May 2024, under the supervision of **Dr. Divyashikha Sethia**.

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other Institute/University. The work has been published/accepted/communicated in SCI/SCI expanded/SSCI/Scopus indexed journal OR peer-reviewed Scopus indexed conference with the following details:

Title of the Paper: **Android App Permission Detector Based on Machine Learning Models**
Author names (in sequence as per research paper): Jaikishan Mohanty, Dr. Divyashikha Sethia
Name of Conference/Journal: IETCIT - 2024
Conference Dates with venue: 1-2nd March 2024, Chandigarh, India
Status of paper (Accepted/Published/Communicated): Accepted
Date of paper communication: January 17, 2023
Date of paper acceptance: February 06, 2023
Date of paper publication: N/A

Jaikishan Mohanty
Roll No. 2K22/SWE/07

Student Roll No., Name and Signature

SUPERVISOR CERTIFICATE

To the best of my knowledge, the above work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere. I further certify that the publication and indexing information given by the students is correct.

Place: Delhi

Dr. Divyashikha Sethia

Date: May 24, 2024

Supervisor Name and Signature



Certificate of Participation

This is to certify that

Dr./Mr./Ms. Jaikishan Mohanty

from Dept. of Soft. Engg., Delhi Technological University, Delhi, India has presented the paper titled
Android App Permission Detector based on Machine Learning Models

to

1st International Conference on "Innovation & Emerging Trends on Computing & Information Technology (IETCIT 2024)" organised by University Institute of Computing, Chandigarh University, Punjab from **1st and 2nd March 2024.**

Dr. Amanpreet Kaur Sandhu
Conference Convenor



Prof. (Dr.) Manisha Malhotra
Conference Chair

Acceptance Notification of Paper ID: 375

Inbox x



IETCIT 2024 <ietcit2024@cumail.in>

to divyashikha, me

Tue, Feb 6, 8:17 PM



Dear Author

Congratulations! On behalf of the Program Committee of IETCIT-2024, we are happy to inform you that the paper title **"Android App Permission Detector based on Machine Learning Models"** with **paper ID 375** has been ACCEPTED for oral presentation in IETCIT-2024 and will be published in CCIS series.

Kindly follow the below mentioned guidelines (strictly), related to preparation of final manuscript, copyright transfer form, payment and final submission. The procedure has been detailed as a five-step process (I)-(V):

(I) **Final Manuscript:** You are requested to give a strict attention to the below mentioned points during preparation of final manuscript to avoid any last minute revert backs from the publisher (Springer).

(a) Please format your paper in the Springer template downloaded from our conference website: <https://www.cuchd.in/ietcit/guideline.php> and the paper template is attached with the mail.

(b) Authors should not mention designation etc. in the name and affiliation part below the paper title. You are requested to mention only the name, college/organization name, city name and mail ids of all the authors.

2) Copyright Form: Kindly fill the copyright form and upload on Google form link.

3) Registration Fee: At least any one of the authors need to register, in the registration link: <https://www.cuchd.in/ietcit/Registrationietcit.aspx>

4) Payments: Kindly fill up the google form after Payment and upload all the following documents in google form: process: <https://forms.gle/n732YuuSXzCREuSVZ>

(a) Final camera-ready paper in Springer format: pdf file. Rename it as "PaperID_M.pdf"

(b) Final camera-ready Springer format source paper: Either in WORD file or in LaTeX. (Mandatory). Rename it as "PaperID_S.doc or PaperID_S.tex"

(c) Copyright form: scan copy or doc. Rename the file as "PaperID_PA.doc".

(d) Upload registration slip as "PaperID.pdf."

(e) Proof of Research scholar/student (if any)

5) Kindly join the whatsapp group: <https://chat.whatsapp.com/KUGliWOkB2B0f6fkJrHutO>

Please complete all the necessary documents by 12th February, 2024. (Strict deadline).

Registration

Already Registered

First Name *

Jaikishan

Last Name *

Mohanty

Phone No. *

7417370013

Email *

jaikishanmohanty@gmail.com

Paper Title *

Android App Permission Detector based on

Paper ID *

375

Abstract *

Android has dominated a significant portion of

Mode of Paper Presentation *

Online

Abstract Id *

IETCIT-15-1033

Total Payment:

5000

INR

Payment Details

Your Registration and Payment is successful and your EmailId is - jaikishanmohanty@gmail.com

Themes of Interest for Submission Include, but are not limited to:

- > Data Science
- > Artificial intelligence
- > Deep Learning
- > Machine Learning
- > Data Forensics & Cyber Security
- > Futuristic Trends in Simulation & Modelling
- > Computational Sciences
- > Advances in Computing
- > IOT
- > Software Engineering

Selected papers will be published in **Scopus indexed** Springer Proceeding

* Communications in Computer and Information Science series (<https://www.springer.com/series/7899>)



NH-05 Chandigarh-Ludhiana Highway, Mohali, Punjab (INDIA)



Important Links

- Conference Theme
- Guideline
- Contact us

Helplines

Phone: 9872257573
Email: ietcit2024@cumail.in

DECLARATION

I hereby certify that the work which is presented in the Major Project-II entitled **Enhancing Android Security: Machine Learning Approaches for App Permissions and Malware Detection** in fulfillment of the requirement for the award of the Degree of Master of Technology in Software Engineering and submitted to the Department of Software Engineering, Delhi Technological University, Delhi is an authentic record of my own, carried out during a period from January to May 2024, under the supervision of **Dr. Divyashikha Sethia**.

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other Institute/University. The work has been published/accepted/communicated in SCI/SCI expanded/SSCI/Scopus indexed journal OR peer reviewed Scopus indexed conference with the following details:

Title of the Paper: **Android App Malware Detection Using Stacking of Machine Learning Algorithms**

Author names (in sequence as per research paper): Jaikishan Mohanty, Dr. Divyashikha Sethia

Name of Conference/Journal: THE 15th INTERNATIONAL IEEE CONFERENCE ON COMPUTING, COMMUNICATION AND NETWORKING TECHNOLOGIES (ICCCNT)

Conference Dates with venue: 24-28th June 2024, IIT Mandi

Status of paper (Accepted/Published/Communicated): Accepted

Date of paper communication: April 15, 2023

Date of paper acceptance: May 22, 2024

Date of paper publication: N/A

Jaikishan Mohanty

Roll No. 2K22/SWE/07

Student Roll No., Name and Signature

SUPERVISOR CERTIFICATE

To the best of my knowledge, the above work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere. I further certify that the publication and indexing information given by the students is correct.



Place: Delhi

Dr. Divyashikha Sethia

Date: May 28, 2024

Supervisor Name and Signature

15th ICCCNT 2024 submission 1486  Inbox x



15th ICCCNT 2024

to me ▾

Wed, May 22, 8:32AM (6 days ago)



"Dear Authors,
Paper ID:1486
Title: Android App Malware Detection using Stacking of Machine Learning Algorithms

Congratulations! Your paper got accepted. In order to include your paper in the presentation schedule, make the following changes otherwise, it will not be considered.

Similarity/Plagiarism Index: 3.9%

1. How feature extraction and selection process is done. Mention the features size that is extracted and mention how many features are selected from it.
2. How the model responds to a new data? How generalizability of the model is achieved?
3. The accuracy of the proposed model is 0.8238. how this prediction accuracy can be increased further?
4. Mention software and system configurations used for generating the results.
5. Include roc plot for proposed classification model.
6. In conclusion, provide details on how the model performance can be further enhanced.

Author affiliation and paper should be in IEEE conference template
(<https://www.ieee.org/conferences/publishing/templates.html>)

***Complete the registration process immediately after receiving this email in order to prepare the presentation schedule (<https://15iccnt.com/register/index.php>).

Similarity/Plagiarism Index should be below 5%.

Kindly update your manuscript based on the above comments and also update the same in the EasyChair login at the same paper ID.
We will inform you of any further updates or changes required, if any.

Best regards,
15th ICCCNT 2024"





To ICCCNT Foundation

₹8,600

Paper Id 1486 Registration Fees

Pay again

Split with friends

✓ Completed

18 Jun 2024, 7:00 pm



ICICI Bank 3518



UPI transaction ID

453648779826

To

••••3777

From: JAIKISHAN MOHANTY (ICICI Bank)

jaikishanmohanty@okicici

Google transaction ID

CICAgPDOxLPYWA



14	Internet of Things	39	Future Technologies	64	Network Based Architectures
15	Biometrics - Click to view tracks	40	Fake news detection	65	Vehicles to everything (V2X) Autonomous car
16	Start-Up	41	QoS provisioning and Architectures	66	AR/VR streaming
17	Electronic Package Topics - Click to view tracks	42	Communication technologies for animals, animal	67	Next Generation Internet
18	Photonics - Click to view tracks	43	Machine Learning for Communications	68	AI/ML techniques for Computer Networks
19	Quantum Photonics	44	Industrial Applications	69	Novel Networking protocols
20	Artificial Intelligence	45	Terahertz Communication	70	Network softwarization for 5G/6G
21	Women in Computing	46	Green Communication Networks	71	Socio-economic impact and regulations for softwarization
22	Data Visualization	47	Multimedia Communication and Software	72	Machine -to- Machine communication networks
23	3D, AutoCAD	48	Cognitive Radio	73	Security/privacy issues
24	AI in Education	49	Information System Security	74	5G /6G In Education
25	Augment Reality, Virtual Reality	50	Communication Theory	75	Cryptography

Publication

15th ICCNT 2024 IEEE Conference proceedings will be published in IEEE Xplorer and indexes in scopus

Qualified paper will be recommended to special issue journals

Venue

The conference will be held in IIT-Mandi, Himachael Pradesh, India

Invited Program committee members and reviewers

If you like to be part of track chair , program committee members, kindly email us with resume

Contact

- Phone . +91 9092939412
- Email . 15th-ieeeconference@iccnt.com