A Model for Android Platform Malware Detection Utilizing Multiple Machine Learning Algorithms

Hussein Al Bazar*[1], Hussein Abdel-Jaber[2], Muawya Naser[3], Arwa Zakaria Hamid[4]
[1,2,4]Faculty of Computer Studies, Arab Open University (AOU), Riyadh, Saudi Arabia
[3] Department of Cybersecurity, Princess Sumaya University for Technology, Amman, Jordan
E-mail: halbazar@arabou.edu.sa, habdeljaber@arabou.edu.sa, m.aldalaien@psut.edu.jo, a.hamid@arabou.edu.sa
*Corresponding author

Keywords: mobile devices, android, malware, security threats, machine learning.

Received: July 3, 2027

In today's technological landscape, the ubiquitous use of mobile devices underscores their critical importance in facilitating daily tasks and enabling a wide array of functionalities, from communication to commerce and entertainment. However, this widespread adoption also brings significant concerns regarding security and privacy, especially with the proliferation of mobile applications capable of accessing sensitive data without explicit user consent. The Android operating system, renowned for its openness and extensive app ecosystem, faces substantial security challenges due to its susceptibility to malware attacks. Malicious software, covertly embedded within seemingly legitimate apps, poses serious threats such as data theft, unauthorized access, and device compromise. This study presents a comprehensive approach to malware detection on the Android platform, utilizing a dataset comprising 4,464 instances, evenly divided between 2,533 labeled as "Malware" and 1,931 labeled as "Benign." The dataset, sourced from real-world Android applications, includes 328 extracted features to enhance detection accuracy. Five machine learning algorithms were evaluated to develop a robust malware detection system: Random Forest, Extra Trees, Logistic Regression, Gradient Boosting, and Support Vector Machine. The performance of these algorithms was rigorously assessed based on accuracy, precision, recall, F1-score, and ROC-AUC. The performance of these algorithms is rigorously evaluated and compared based on accuracy, precision, recall, and F1-score. The results reveal that the Logistic Regression algorithm achieved the highest accuracy at 97.31%, outperforming the other models. Specifically, Random Forest achieved 96.64%, Extra Trees 96.08%, Gradient Boosting 96.19%, and Support Vector Machine 96.75%. These findings suggest that Logistic Regression is particularly effective in identifying Android malware within this dataset, offering a reliable solution for enhancing mobile security. This research benchmarks these results against prior models utilizing different machine learning approaches and provides concrete insights into the most effective methodologies for mitigating Android malware threats. By advancing detection capabilities through sophisticated machine learning techniques, this study contributes to ongoing efforts to safeguard mobile device users from evolving cybersecurity threats, underscoring the critical role of data-driven models in enhancing the security and privacy of Android platforms.

Povzetek: Študija uvaja model za zaznavanje zlonamerne programske opreme na Android platformi z uporabo mnogoterih algoritmov strojnega učenja - tako učinkovito zmanjšuje varnostna tveganja na mobilnih napravah.

1 Introduction

As technology advances, mobile devices are increasingly relied upon by users [1, 2]. The contemporary era is seeing a rapid proliferation of mobile devices and their associated apps, driven by their seamless functionality and ongoing improvements in smartphone technology [3]. Mobile phones have become omnipresent, enabling a multitude of daily tasks such as calling, web browsing, online banking, social networking, e-commerce, gaming, photography, and app usage [4-6]. Furthermore, the vast array of capabilities offered by mobile devices and the expanding range of user activities have raised significant concerns regarding device security and personal privacy [7, 8].

Mobile applications installed on a device can utilize various features and capabilities, such as the integrated GPS for location tracking, the camera, and the microphone. This extensive access to the device's hardware and sensors enables installed apps to collect data without the user's explicit consent or awareness [5]. According to a study by the International Association of Mobile Operators (GSMA), as of 2023, 5.6 billion people, constituting 69% of the global population, were subscribed to mobile services. Additionally, 58% of the world's population accesses mobile Internet via smartphones, totaling 4.7 billion users [9].

Several operating system platforms are accessible for mobile phones in the marketplace, such as Windows Mobile, iOS, and Android OS, the most prevalent mobile phone operating system [10-12]. Android OS is recognized for its high level of customization and open-

source nature, supporting a wide range of devices, including smartphones, smartwatches, television sets, and projectors. This flexibility allows Android OS to adapt to diverse hardware platforms and form factors, boasting a repository of over 2.6 billion apps available on the Google Play Store by 2023 [5, 13]. However, the open-source framework of Android OS, combined with lax security vetting of marketplace applications, exposes Android devices to heightened vulnerability against malicious attacks [11]. Additionally, the absence of built-in tools within Android OS to preemptively detect malware in applications before installation leaves users unable to ascertain the presence of potentially harmful components [4, 6].

Malware refers to software designed to disrupt or impair computer or mobile applications, gather sensitive information, or carry out malicious activities [2, 4, 10]. It can be covertly packaged with legitimate applications, or its malicious functionalities can be concealed within an otherwise benign-looking app, posing potential threats to user data and device integrity [2, 4]. Malware's main objectives include impairing applications, stealing sensitive data like credit card details or login credentials, accessing personal information, and executing harmful actions [11, 12]. Cybercriminals have increasingly targeted the Android platform by embedding sophisticated malicious code in applications, challenging security providers to effectively detect and counter such threats [12, 14]. Furthermore, malicious applications often bypass existing security measures, leaving users unaware if an installed app might operate maliciously and engage in harmful activities undetected [5].

The conventional method for identifying malware is signature-based detection, where a unique identifier or signature is extracted from the application and compared with a database of known malware signatures [15]. However, this approach has a significant drawback: even slight modifications to the malware, such as changes in code lines, instructions, or keywords, can alter the signature sufficiently to evade detection by anti-malware software [15]. In addition to signature-based detection, machine learning offers two primary approaches for malware detection: supervised and unsupervised methods [16]. Furthermore, various techniques for analyzing and detecting Android malware fall into categories such as static, dynamic, or hybrid analysis [13, 14]. These methods aim to extract key feature sets used by machine learning algorithms for subsequent malware detection [14]. Nonetheless, a significant drawback of these approaches is their complexity and high computational demands, owing to their reliance on numerous features [13, 15].

Android malware poses a significant security risk by exploiting vulnerabilities in Android devices, potentially leading to financial losses and unauthorized access to sensitive personal information. As these malware attacks become more prevalent, there is an increasing demand for robust detection methods to protect users from these malicious threats [16, 17].

This study proposes an Android-based platform using machine learning models that employ various algorithms: Random Forest (RF), Extra Trees (ET), Logistic Regression (LR), Gradient Boosting (GB), and Support Vector Machine (SVM), for application malware detection. The objective is to determine which algorithm provides the best results in terms of accuracy, precision, recall, and F1-score. Additionally, the study compares its findings with a previously developed model in [18] utilizes the same dataset using different machine learning algorithms to determine which one has better accuracy classification report results. This research aims to determine the machine learning algorithm that produces the best classification results among those evaluated. Furthermore, it seeks to identify whether the developed model or previous models utilizing different machine learning algorithms demonstrate superior accuracy results.

The paper is structured as follows: Section 2 reviews existing related works on models for detecting Android malware. Section 3 describes the developed machine learning model in detail. Section 4 provides an overview of the dataset used in this research. Section 5 presents the achieved classification results and a discussion of the compared algorithms. Section 6 introduces the classification results and discussion between the proposed model and a previous model. Conclusions and future work based on the findings of the study are revealed in Section 7.

2 Related work

This section discusses the most recent studies on employing machine learning models for detecting malware on the Android platform.

In [1], a method for detecting Android malware is introduced based on aggregating vulnerable features. This approach uses static analysis to link each API call with specific features, which are combined to determine their frequency. Several machine learning classifiers were tested, with RF showing the highest effectiveness, achieving an ROC-AUC score of 98.87%, particularly for obfuscated malware. The study also employs Non-negative Matrix Factorization (NMF) to streamline the model, demonstrating its scalability by reducing the feature set by 75.9% while maintaining a strong ROC-AUC score of 95.67%. Moreover, in [2]. The researchers propose a machine-learning approach using the SVM algorithm to detect Android malware. The study emphasizes static analysis of API calls made by Android apps. The dataset includes malicious and benign applications accessing system resources via Android API calls. Comparative analysis against other methods shows exceptional performance, achieving an overall accuracy of 99.75% [2]. Furthermore, in [8], a static-based classification approach is introduced for Android malware detection using permissions and API calls. The study employs SVM, K-nearest neighbors (KNN), and Naive Bayes (NB) classifiers on the "CIC InvesAndMal2019" dataset. Results indicate that SVM achieves the highest average accuracy rate of 94%, followed by KNN at 93% and NB at 84%.

The study [19] presents an effective method for feature extraction from Android APKs using static analysis. The

research utilizes diverse features such as API calls, intents, permissions, and command signatures. Extracting these features employs two well-known datasets, Drebin and Malgenome. Machine learning classifiers, including SVM, MVC, and RF, are trained on these features. Additionally, Principal Component Analysis (PCA) is used to optimize feature sets and train models. Results show that RF achieves % the highest accuracy of 96.27% with the Drebin permission feature combination. Meanwhile, SVM with PCA surpasses other classifiers, reaching a peak accuracy of 97.63% using the Malgenome permission combination [19]. Additionally, in [7], a novel approach for Android malware detection is proposed based on the coexistence of features. A new dataset is created, integrating co-occurring permissions and API calls at varying combination levels. The frequent pattern growth method extracts the most relevant coexisting features. Android APK samples from Drebin, Malgenome, and MalDroid2020 datasets were utilized to generate these new datasets. Using the Random Forest algorithm with coexisting permissions features, the approach achieves a maximum accuracy of 98%. Specifically, the Malgenome dataset outperforms the state-of-the-art accuracy of 87%, reaching 98%. Furthermore, compared to the state-of-the-art 93% accuracy using the Drebin dataset, the proposed method achieves 95% accuracy [7].

In a recent study conducted by [5], the effectiveness of various machine-learning approaches for classifying Android malware using the Drebin dataset was evaluated. The researchers analyzed a dataset comprising 123,453 benign and 5,560 malware applications sourced from Drebin. Their primary goal was to assess the performance of several classifiers and determine the key features influencing their decision-making. The study found that SVM achieved the highest accuracy, precision, and recall metrics among the classifiers tested, which included RidgeReg, RF, LassoReg, BNB, 1-ANN, and ANN. In contrast, RidgeReg was identified as the optimal choice due to its model complexity, optimization capabilities, and ability to fine-tune [5].

In [20], the effectiveness of various machine-learning models for detecting Android malware is investigated. The study employs SMOTE normalization for numerical features and Principal Component Analysis (PCA) to enhance accuracy. A robust multi-category classification system for Android malware is also introduced based on the Light Gradient Boosting Model (GBM). This model categorizes malware into five classes: Adware, Banking Malware, SMS Malware, Mobile Riskware, and Benign apps. Results highlight that the Light GBM algorithm achieves the highest F1-score of 95.47% and the best accuracy of 95.49% [20].

In [21], the research explores static, dynamic, and hybrid analyses for detecting malware. The dataset includes benign, malware, and Greyware applications. Various classifiers such as XGBoost, Gradient Boosting, Decision Tree, and Random Forest are employed. The study highlights the effectiveness of static features (specifically permissions) and dynamic features (activity repetition). Results indicate that accuracy rates for static, dynamic, and hybrid analyses are consistently above 94%, with static analysis particularly noted for its cost-effectiveness in classification tasks. Furthermore, in [22], BrainShield has introduced a hybrid malware detection model designed to defend against attacks on Android devices. Trained on the Omnidroid dataset, BrainShield integrates static, dynamic, and hybrid neural networks. The results indicate that the model achieved accuracies of 92.9% with static analysis, 81.1% with dynamic analysis, and 91.1% with hybrid (combined static and dynamic) analysis.

In [23], a method is introduced for detecting Android malware by analyzing correlations among abstracted API calls made at the method level within applications. These API calls are grouped into transactions to establish behavioral semantics for each application. The system characterizes app behavior by calculating the confidence of association rules between these calls. Machine learning techniques are then applied to integrate distinct behavioral patterns of malicious and benign apps, creating an effective detection system. Results show strong performance, achieving 96% accuracy and 98% F-measure on the Drebin and AMD datasets, demonstrating competitive accuracy and efficiency [23]. Moreover, research introduces PermDroid, a framework designed to enhance the effectiveness of machine learning-based Android malware detection through feature selection techniques [13]. Initially, PermDroid applies statistical methods such as t-test and logistic regression to identify the most relevant features distinguishing malware from benign apps. It then refines the feature set through regression and correlation analyses. The optimized subset of features is utilized to build malware detection models using three ensemble approaches: homogeneous, heterogeneous, and linear ensembles. Experimental findings demonstrate that PermDroid's feature selection strategy enables models like DNN and NDTF to achieve a high malware detection accuracy of 98.8%, surpassing previous frameworks [13].

An automated approach for detecting Android malware using the Optimal Ensemble Learning Approach for Cybersecurity is introduced in [24]. The primary objective of this technique is to automate the classification and identification of Android malware. The Android malware detection process employs ensemble learning with LS-SVM, KELM, and RRVFLN models. Additionally, parameter tuning based on HPO enhances malware detection results. The simulation results demonstrate the superiority of the proposed technique over existing approaches [24]. A malware detection method called CogramDroid is introduced in [25], which relies on opcode ngrams. This method categorizes applications by analyzing the relative frequency patterns of opcode ngrams, employing the concept of word cooccurrence from natural language processing. The result shows that using three grams and seven core opcodes, CogramDroid achieved an accuracy rate of 96.22% and an F1-score of 96.69% [25]. Additionally, this work presents a framework that uses reverse-engineered Android app features and advanced machine learning to identify security vulnerabilities. The researchers developed a model incorporating cutting-edge static analysis methods

and leveraging extensive malware datasets [25]. Additionally, the study employed an ensemble learning strategy, integrating multiple machines learning algorithms, including AdaBoost, SVM, Decision Trees, KNN, NB, and RF. This ensemble approach was used to enhance the performance and accuracy of the vulnerability detection model. The proposed framework was evaluated across three distinct datasets, and the experimental results demonstrate an accuracy of 96.24% in detecting malware extracted from Android applications using the AdaBoost machine-learning technique [12].

In [26], an Android malware detection method utilizing multiple machine learning algorithms is introduced. The study explores features extracted from Android app binaries, including permissions, API calls, system calls, code segments, and structural attributes. Results indicate that SDL achieves the highest accuracy rate of 99.3%, followed closely by SVM at 98.69%. Similarly, in [11], a method leveraging Java application permissions and APIs for Android malware detection is presented. RF, SVM, NB, and DT machine learning algorithms are applied, with RF achieving the highest accuracy rate of 96.2% among the tested techniques. These studies underscore the efficacy of machine learning approaches in identifying and categorizing malicious software on Android devices, demonstrating significant advancements in malware detection capabilities.

Researchers explored Android malware detection using a hybrid approach combining machine learning with genetic algorithms. Their study involved deploying nine machine-learning algorithms on a dataset containing 5,000 benign Android apps and 2,500 malware samples, each characterized by 1,104 static features. Notably, they integrated a genetic algorithm-based feature selection technique to pinpoint the most informative attributes from the dataset. Experimental results showcased that this genetic algorithm-based strategy significantly boosted overall malware detection performance, especially in terms of time efficiency, compared to traditional machine learning models lacking feature selection [17]. Furthermore, in a study conducted by [4], the MobiPCR malware detection system for the Android platform is introduced, incorporating a cloud-based architecture and an edge computing model. This innovative design utilizes machine learning frameworks to achieve efficient and accurate malware detection while reducing computational and power demands on mobile devices. The core functionality of MobiPCR involves uploading new application binaries to a cloud-hosted detector, which performs code analysis and applies advanced machine-learning techniques to detect potential malicious components [4].

In [27], a parallel machine learning-based method is introduced for early detection. Using a custom analysis tool, this approach employs diverse classifiers and extracts static features such as API-related features, app permissions, and OS/framework commands from Android app APK files. These features train multiple machine learning models in a parallel classification setup, including DT, Simple Logistic, NB, PART, and RIDOR. Experimental results demonstrate that the PART algorithm achieves superior detection capability and accuracy, performing at 95.8% and 96.3%, respectively, outperforming other algorithms in the study [27]. In contrast, [6] proposes an approach for identifying malware in Android applications using image-based feature extraction. This method converts files from Android app source code into grayscale images. It extracts various image features such as SIFT, SURF, KAZE, ORB, color histograms, Haralick texture measures, and Hu moments. These features train machine learning models, including RF, AdaBoost, and GB. Testing across three datasets containing 9,700 samples shows that RF, AdaBoost, and GB classifiers achieve high accuracies ranging from 95.78% to 98.75% when utilizing global image features. These studies highlight advanced methodologies in Android malware detection, combining sophisticated feature extraction techniques with powerful machine-learning algorithms to enhance accuracy and efficiency in detecting malicious applications.

A method to identify repackaged Android malware adopts a comprehensive approach that examines the app's internal code structure, behavioral patterns, and dependency relationships. This involves segmenting the code into graphs, identifying vulnerabilities at both class and method levels, and utilizing an evolving genetic algorithm to optimize a feature set. This optimized feature set is then used to train machine learning models efficiently to detect repackaged malware. Experimental findings demonstrate that by using lower-dimensional feature sets, classifiers such as SVM and Neural Networks consistently achieve classification accuracies exceeding 90-91%, effectively streamlining the training process while maintaining high accuracy levels [14]This approach highlights the integration of advanced techniques in code analysis and machine learning to combat repackaged Android malware effectively.

Numerous research studies have conducted thorough and systematic reviews on the application of machine learning techniques for Android malware detection, as evidenced by studies such as [15, 16, 28]. These reviews aim to provide a comparative analysis of different methods for detecting Android malware and discuss the evaluation metrics employed to measure their effectiveness. The overarching goal is to offer a comprehensive overview of the current state-of-the-art in Android malware detection, highlighting the strengths and limitations of various machine learning-based approaches. By assessing these methods using standardized metrics, these studies aim to assist researchers and practitioners in making well-informed decisions when selecting malware detection solutions tailored to their specific requirements [15, 16, 28].

Researchers have extensively explored deep learning techniques for Android malware detection, showcasing a variety of innovative approaches and their respective achievements. For instance, Alzaylaee et al. introduced DL-Droid, a dynamic deep-learning system that analyzes Android applications and achieved detection rates up to 97.8% using dynamic features alone and 99.6% when incorporating dynamic and static features [29]. Another system, MAPAS, utilizes convolutional neural networks

to analyze API call graphs, achieving an accuracy rate of 91.27% [30]. Deep learning algorithms such as LSTM and MLP have also been effectively employed, demonstrating malware detection accuracies exceeding 99% across various datasets [31]. Moreover, a convolutional neural network-based method proposed in [32] achieved an impressive accuracy rate of 99.92% by optimizing parameters such as filter sizes, training epochs, learning rates, and network configurations using the Drebin dataset. Additionally, studies like [33] explored GRU-based methods while [34] introduced MobiTive, an efficient Android malware detection system leveraging customized deep neural networks for responsive real-time detection on mobile devices. These advancements underscore the growing effectiveness of deep learning in enhancing Android malware detection capabilities. Table 1 summarizes some recent studies that have discussed the method used, the dataset used, and the performance matrices based on accuracy, precision, recall, and F1-score.

Table 1: A summary of recent android malware detection solutions.

Ref	Year	Method (s)	Dataset	Accuracy	Precision	Recall	F1-Score
[26]	2024	SVM	Android applications collected data.	98.69	98.7	98.7	98.7
		CNN		97.53	97.6	97.5	97.5
		AE		95.33	95.3	95.3	95.3
		DBN		97.7	97.7	97.7	97.7
		ELM		98.2	98.2	98.2	98.2
		SDL		99.34	99.3	99.3	9.3
[3]	2023	Naïve Bayes	APK	85.1	-	-	-
		KNN		87.2	-	-	-
		SVM		91.4	-	-	-
		DBN-GRU		97.9	-	-	-
[4]	2023	Hamming DES	Collected from multiple sources. Application Centers Drebin Ashish Bhatia	94.5	-	-	-
		Voting		70.4	-	-	-
		Naïve stacking		87.7	-	-	-
		MobiPCR		99.1	-	-	-
[20]	2023	LGB	CICMalDroid2020	94.80	-	-	94.81
		RF		94.11	-	-	94.12
		ET		93.74	-	-	93.77
		KNN		84.66	-	-	84.71
		SVM		45.88	-	-	45.53
[7]	2023	RF Best-achieved accuracy.	A new dataset was created from Drebin, Malgenome, and MalDroid2020 APK samples. RF results were shown with varying numbers of selected features.	98.0	-	-	-
[8]	2022	SVM	CICInvesAndMal2019 Permission features	94.36	95.9	82.6	88.8
		KNN		93.42	91.1	83.7	87.3
		NB		84.33	97.4	63.0	70.8
[19]	2022	SVM	Drebin (D) Malgenome (M) With the usage of Principal Component Analysis (PCA).	D 98.19 M 98.84	D 98.0 M 99.0	D 97.0 M 97.0	D 98.0 M 98.0
		MVC		D 97.65 M 98.24	D 98.0 M 1.0	D 95.0 M 94.0	D 97.0 M 96.0
		RF		D 97.15 M 97.78	D 98.0 M 1.0	D 94.0 M 95.0	D 96.0 M 97.0
[21]	2022	GB	Palo Alto Networks With static feature results	99.5	99.4	99.7	99.6
		XGBoost		99.5	99.4	99.7	99.6
		DT		99.4	99.4	99.7	99.4
		RF		94.6	95.1	99.4	97.2
[17]	2021	J48	Andro-AutoPsy	96.8	-	-	95.2
		RF		97.8	-	-	96.6
		Decision Table		94.6	-	-	91.6
		MLP		98.1	-	-	97.1
		NB		69.1	-	-	91.0
		SVM		96.4	-	-	96.4
		LR		96.3	-	-	94.4

		AdaBoost		88.4	-	-	81.7
		KNN		97.2	-	-	95.7
[1]	2020	LR	Drebin	91.86	94.77	84.72	89.47
		SVM		93.35	90.88	93.06	91.95
		RF		93.77	99.80	84.72	91.73
		KNN		93.15	89.37	94.44	91.84
[2]	2020	SVM Best-achieved accuracy.	Collected from an Android environment including Google Play, Amazon AppStore, APKP, AMD, and Drebin	99.75	99.54	99.97	-
[11]	2020	SVM	Android MSA	84.5	-	-	81.9
		ID3		80.7	-	-	79.9
		NB		91.4	-	-	88.7
		RF		96.2	-	-	94.9
[23]	2019	KNN	Drebin (D) AMD	D 92.0 96.0	D 90.0 95.0	D 96.0 97.0	D 93.0 96.0
		RF		D 96.0 98.0	D 97.0 99.0	D95.0 98.0	D 96.0 98.0
		SVM		D94.0 97.0	D 94.0 97.0	D 94.0 97.0	D 94.0 97.0

To conclude, Android malware detection, while advanced, still faces significant challenges that highlight the need for further research. Many existing methods heavily depend on static or dynamic analysis, each with limitations such as vulnerability to code obfuscation or high computational demands. These techniques cannot often generalize effectively across the diverse and rapidly evolving Android environment, resulting in issues with both accuracy and scalability. Additionally, while machine learning approaches are increasingly being utilized, they often rely on single models that may not fully capture the wide range of malware behaviors, leading to less-than-optimal detection rates. Existing studies overlook the potential benefits of combining multiple machine learning algorithms to enhance detection performance. A significant gap in existing studies is the narrow focus on accuracy as the sole metric for evaluation, overlooking other crucial aspects like robustness and adaptability. To address these gaps, this research introduces a model that employs a combination of multiple machines learning algorithms, enhancing detection robustness and accuracy. This approach mitigates the weaknesses of single-method strategies and offers a more comprehensive defense against sophisticated malware, representing a crucial advancement in the field.

3 Malware detection model

A model for detecting malware on the Android platform has been developed using several machine learning algorithms such as Support Vector Machine, Random Forest, Extra Trees, Logistic Regression, and Gradient Boosting. The model's workflow is depicted in Figure 1. Initially, the dataset utilized was obtained from the Kaggle website [35], which serves as a platform providing a variety of datasets for machine learning projects.

The preprocessing stage involved several critical steps to ensure data readiness. The encoding of categorical variables in the dataset is facilitated using the LabelEncoder class. This method transforms categorical data into numerical labels, assigning a unique integer to each category within a column. Implemented within the custom Encoder class, each categorical feature transforms individually, ensuring compatibility with subsequent machine learning algorithms that require numerical input rather than categorical labels. LabelEncoder class encodes the values of target variables into values between 0 and the number of class values– 1.

To address class imbalance within the dataset, the RandomOverSampler class is applied. This method randomly duplicates instances of the minority class (malware) and appends them to the dataset until it achieves a balanced representation of both classes (malware and benign). By oversampling the minority class within the training set after the initial split using train_test_split, the model training process is fortified against bias towards the majority class, thereby enhancing its ability to generalize effectively.

After balancing the dataset, feature selection is implemented using the SelectFromModel class. SelectFromModel can be applied with an estimator to give importance to every feature by using a particular attribute like coef_, feature_importances_, or importance_getter callable beyond fitting [36]. When the importance of the values of features is less than the given threshold value, these features are not significant and are deleted [36]This approach involves selecting the most important features based on the importance of the fitted model's features. For instance, for the algorithms used in the developed model, the SelectFromModel is integrated into each pipeline after preprocessing. This technique helps improve model efficiency and generalization by focusing on the most discriminative features while discarding less relevant ones. By reducing the dimensionality of the dataset to these influential attributes, overfitting is mitigated, and

model performance is enhanced, contributing to more accurate predictions in detecting and classifying Android malware.

Finally, the dataset undergoes partitioning into training and testing subsets utilizing the StratifiedShuffleSplit cross-validation method. StratifiedShuffleSplit offers indices for train/test aiming at dividing data into sets for train/test [36]. This strategic approach involves randomly shuffling and splitting the dataset while maintaining the original distribution of classes, ensuring that each split retains the same proportion of Android malware and benign samples. By configuring n_splits=5, the dataset is segmented into five distinct training and testing sets across multiple iterations. This methodology is pivotal as it guarantees that the machine learning models are trained on diverse yet balanced subsets of data, facilitating robust evaluation and validation. Consequently, the models are rigorously assessed for their ability to generalize to unseen data, thereby bolstering their reliability and effectiveness in classifying Android malware.

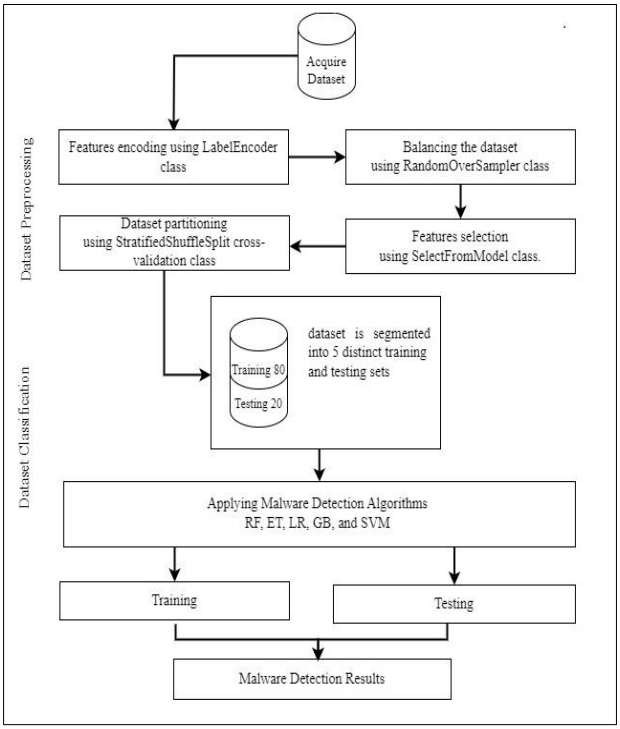

Figure 1: The machine learning malware detection model.

4 The used dataset

The dataset obtained from Kaggle, specifically tailored for detecting Android malware [35], encompasses 328 features extracted from Android applications. These carefully curated features aim to cover various facets of app behavior, assisting machine learning models in effectively identifying patterns linked to malware [35]. With a total of 4,464 instances, the dataset is evenly divided between 2,533 instances labeled as "Malware" and 1,931 instances labeled as "Benign". The decision to utilize this dataset was motivated by its comprehensive nature, offering researchers a trove of information for malware detection and analysis on the Android platform. As part of the preprocessing steps, it was observed that the data suffered from imbalance, with more instances of "Malware" compared to "Benign" applications. To address this limitation, a RandomOverSampler technique was employed during preprocessing to balance the dataset. This approach aimed to mitigate the effects of data imbalance and ensure that the machine learning models are trained on a more representative dataset, enhancing the robustness and accuracy of the analysis.

5 Results and discussion of the compared algorithms

This section presents the results of the developed model, which relies on machine learning algorithms for Android malware detection. The model utilizes Random Forest, Extra Trees, Logistic Regression, Gradient Boosting, and Support Vector Machine algorithms. Within the framework, these algorithms are compared across multiple parameters, including accuracy, precision, recall, and F1-Score. The objective is to determine which algorithm demonstrates the most favorable outcomes among those employed.

As shown in Figure 2, the performance of the RF model was evaluated using a comprehensive set of metrics, including precision, recall, F1-score, and accuracy on both training and testing datasets. The RF model demonstrated robust capabilities with a precision of 0.96 and a recall of 0.96 for the Benign class, indicating that 96% of instances predicted as benign were correct and accurately identified 96% of actual benign instances. For malware detection, the model achieved a precision of 0.97 and a recall of 0.97, highlighting its high accuracy in predicting malware instances. The balanced performance of the RF model is further underscored by the F1-scores of 0.96 for the Benign class and 0.97 for the Malware class. These scores reflect the model's ability to effectively balance between correctly identifying positives and minimizing false positives, which is crucial for reliable malware detection systems. In terms of generalization, the RF model exhibited strong performance on previously unseen test data, achieving an accuracy of 96.64%. This indicates that the model successfully applied its learned patterns from the training set to new instances, demonstrating robust performance without overfitting.

	precision	recall	f1-score	support
Benign	0.96	0.96	0.96	377
Malware	0.97	0.97	0.97	516
accuracy			0.97	893
macro avg	0.97	0.97	0.97	893
weighted avg	0.97	0.97	0.97	893
Accuracy Train: 0.9871		Accuracy Test: 0.9664		

Figure 2: Random Forest malware classification report.

Referencing Figure 3. The Extra Trees classifier demonstrated robust capabilities, achieving a precision of

0.95 and a recall of 0.96 for the Benign class. These metrics indicate that 95% of instances predicted as benign were accurate, and the classifier correctly identified 96% of actual benign instances. For malware detection, the classifier achieved a precision of 0.97 and a recall of 0.96, highlighting its high accuracy in predicting malware instances. The balanced performance of the Extra Trees classifier is further underscored by the F1-scores of 0.95 for the Benign class and 0.97 for the Malware class. These scores illustrate the classifier's ability to effectively manage the trade-off between correctly identifying and minimizing false positives, which is essential for robust malware detection systems. In terms of accuracy, the Extra Trees classifier achieved an impressive overall accuracy of 96.08% on the test dataset.

	precision	recall	f1-score	support
Benign	0.95	0.96	0.95	377
Malware	0.97	0.96	0.97	516
accuracy			0.96	893
macro avg	0.96	0.96	0.96	893
weighted avg	0.96	0.96	0.96	893
Accuracy Train: 0.9784		Accuracy Test: 0.9608		

Figure 3: Extra trees malware classification report.

For the Logistic Regression Algorithm, as illustrated in Figure 4. The Logistic Regression classifier exhibited robust performance metrics across benign and malware classes. The Benign class achieved a precision of 0.98 and a recall of 0.96, indicating that 98% of instances predicted as benign were accurate and correctly identified 96% of actual benign instances. In detecting malware, the classifier achieved a precision of 0.97 and a recall of 0.98, highlighting its high accuracy in identifying malware instances. The balanced performance is further highlighted by the F1-scores of 0.97 for the Benign class and 0.98 for the Malware class, demonstrating its ability to effectively balance between correctly identifying positives and minimizing false positives, which is crucial for reliable malware detection systems. Moreover, achieving an overall accuracy of 97.31% on the test dataset underscores the classifier's robustness in generalizing its learned patterns to new instances without overfitting.

	precision	recall	f1-score	support
Benign	0.98	0.96	0.97	377
Malware	0.97	0.98	0.98	516
accuracy			0.97	893
macro avg	0.97	0.97	0.97	893
weighted avg	0.97	0.97	0.97	893
Accuracy Train: 0.9695		Accuracy Test: 0.9731		

Figure 4: Logistic regression malware classification report.

Referencing Figure 5, the Gradient Boosting classifier demonstrated robust performance across metrics crucial for Android malware detection. The Benign class achieved a precision of 0.96 and a recall of 0.95, indicating that 96% of instances predicted as benign were correct and accurately identified 95% of actual benign instances. In detecting malware, the classifier achieved a precision of 0.96 and a recall of 0.97, highlighting its high accuracy in identifying malware instances. The balanced performance is further illustrated by the F1-scores of 0.95 for the Benign class and 0.97 for the Malware class, showcasing the classifier's capability to effectively balance between correctly identifying positives and minimizing false positives, essential for robust malware detection systems. Moreover, achieving an overall accuracy of 96.19% on the test dataset underscores the classifier's ability to generalize learned patterns to new instances without overfitting.

	precision	recall	f1-score	support
Benign	0.96	0.95	0.95	377
Malware	0.96	0.97	0.97	516
accuracy			0.96	893
macro avg	0.96	0.96	0.96	893
weighted avg	0.96	0.96	0.96	893
Accuracy Train: 0.9636		Accuracy Test: 0.9619		

Figure 5: Gradient boosting malware classification report.

As illustrated in Figure 6. The Support Vector Machine classifier exhibited robust performance metrics crucial for Android malware detection. The Benign class achieved a precision of 0.98 and a recall of 0.95, indicating that 98% of instances predicted as benign were correct and accurately identified 95% of actual benign instances. In detecting malware, the classifier achieved a precision of 0.96 and a recall of 0.98, highlighting its high accuracy in identifying malware instances. The balanced performance is further underscored by the F1-scores of 0.96 for the Benign class and 0.97 for the Malware class. This demonstrates the classifier's ability to effectively balance between correctly identifying positives and minimizing false positives, which is crucial for robust malware detection systems. Additionally, achieving an overall accuracy of 96.75% on the test dataset illustrates the classifier's capability to generalize learned patterns to new instances without overfitting.

	precision	recall	f1-score	support
Benign	0.98	0.95	0.96	377
Malware	0.96	0.98	0.97	516
accuracy			0.97	893
macro avg	0.97	0.96	0.97	893
weighted avg	0.97	0.97	0.97	893
Accuracy Train: 0.9714		Accuracy Test: 0.9675		

Figure 6: Support vector machine malware classification report.

According to the discussion mentioned above, it can be shown that across the evaluated machine learning, including RF, ET, LR, GB, and SVM, each algorithm demonstrated robust capabilities in distinguishing between benign and malware applications. RF and LR exhibited high precision and recall for both benign (RF: 0.96, 0.96; LR: 0.98, 0.96) and malware (RF: 0.97, 0.97; LR: 0.97, 0.98) classes, reflecting their strong accuracy in classification tasks. Extra Trees and Gradient Boosting also performed well with precision-recall metrics (ET: benign 0.95, 0.96; malware 0.97, 0.96; GB: benign 0.96, 0.95; malware 0.96, 0.97) and balanced F1-scores (ET: benign 0.95, malware 0.97; GB: benign 0.95, malware 0.97), indicating effective trade-offs between correct identifications and false positives. SVM showed similarly strong performance (benign 0.98, 0.95; malware 0.96, 0.98) with balanced F1-scores (benign 0.96, malware 0.97). Logistic Regression stood out with the highest overall accuracy of 97.31%, closely followed by SVM (96.75%), RF (96.64%), ET (96.08%), and GB (96.19%) on the test dataset. These results suggest that while each classifier excels in specific metrics, Logistic Regression demonstrates the most favorable outcomes overall for Android malware detection tasks, balancing precision, recall, and accuracy effectively.

Moreover, the study referenced in [18] employs identical data sourced from the Kaggle website and applies various machine-learning algorithms for Android malware detection. This study compares its developed model with the one in [Ref], specifically focusing on Random Forest, Extra Trees, Logistic Regression, Gradient Boosting, and SVM algorithms using accuracy as the primary metric. Accuracy holds significant importance in research, often serving as a pivotal indicator to assess the efficacy of malware detection techniques in scholarly articles.

The accuracy, precision, recall, and F1-score results for Random Forest, Extra Trees, Logistic Regression, Gradient Boosting, and SVM are provided in Figures 7-11, respectively. In addition, these results, besides macro average and weighted average results for the same order of algorithms, are shown in Table 2.

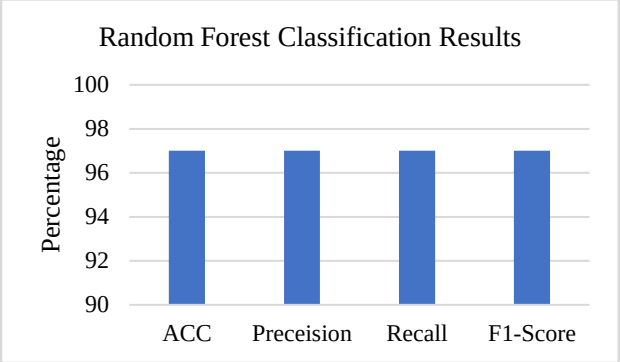

Figure 7: Random Forest malware classification results.

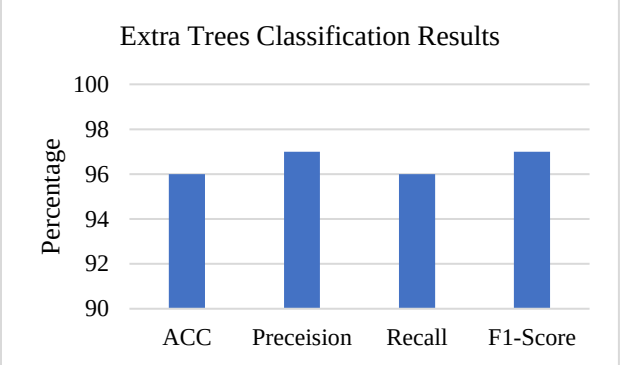

Figure 8: Extra trees malware classification results.

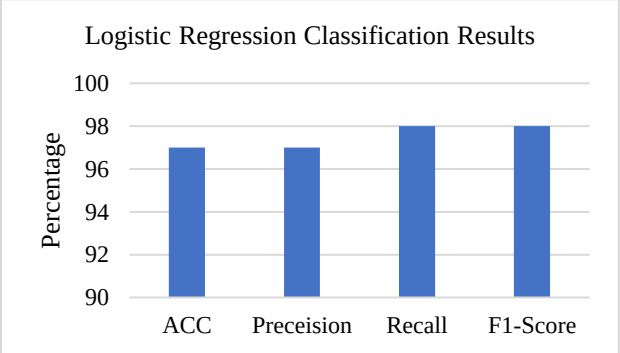

Figure 9: Logistic regression classification results.

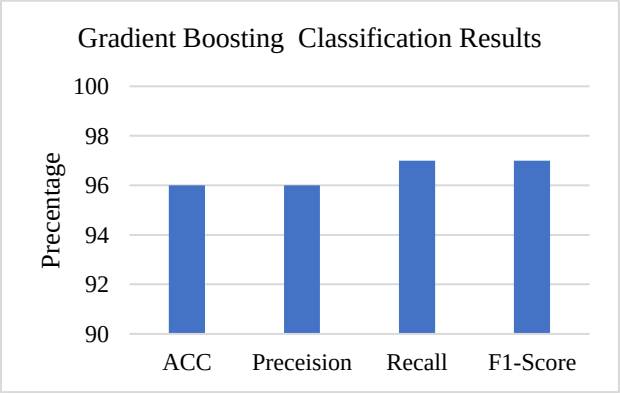

Figure 10: Gradient boosting classification results.

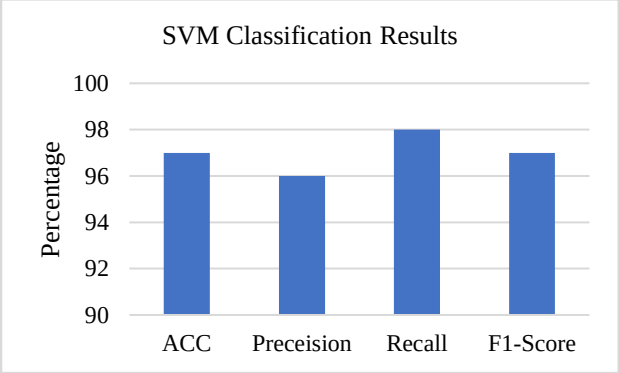

Figure 11: SVM malware classification results.

Table 2: The used ML algorithms malware classification results.

ML Algorithm	RF	ET	LR	GB	SVM
Accuracy	0.97	0.96	0.97	0.96	0.97
Precision	0.97	0.97	0.97	0.96	0.96
Recall	0.97	0.96	0.98	0.97	0.98
F1-score	0.97	0.97	0.98	0.97	0.97
macro avg	0.97	0.96	0.97	0.96	0.97
weighted avg	0.97	0.96	0.97	0.96	0.97

The results of accuracy, precision, recall, and F1-score of the compared algorithms are shown in Figure 12. Random Forest, Logistic Regression, and SVM offer similar accuracy results, and these results are better than the accuracy results for Extra Trees and Gradient Boosting. Both Extra Trees and Gradient Boosting provide similar accuracy results. Random Forest, Extra Trees, and Logistic Regression present similar precision results, which are higher than those for Gradient Boosting and SVM. The precision results for Gradient Boosting and SVM are similar. The recall results for Logistic Regression and SVM are similar and the highest. In addition, Random Forest and Gradient Boosting have similar recall results, which are better than the recall result for Extra Trees. Logistic Regression provides the best F1-score result. Random Forest, Extra Trees, Gradient Boosting, and SVM have similar F1-score results.

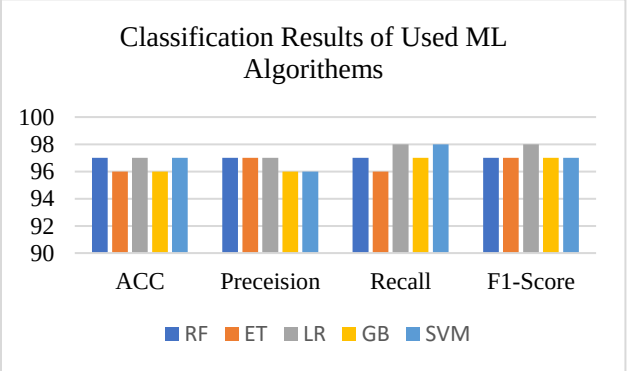

Figure 12. Classification results of the compared algorithms.

The Receiver Operating Characteristic (ROC) curve is presented in Figure 13. The ROC curve represents the classifier performance on dissimilar classification thresholds [37]. The figure of the ROC curve shows the True Positive Rate versus the False Positive Rate.

The Receiver Operating Characteristic Area Under the Curve (ROC AUC) is a result that summarizes the classifier's performance for all potential classification thresholds [37]. ROC AUC results can be accomplished by gauging the area under the ROC curve [37]. ROC AUC result reveals how the classifier differentiates between negative and positive classes. As long as the ROC AUC result is higher, the performance is more satisfactory. The perfect model has an AUC result equal to 1 [37]. Figure 13 shows that the AUC results for all compared algorithms are similar, and these results are 0.99. These results indicate that the model's performance using the compared algorithms is near perfect.

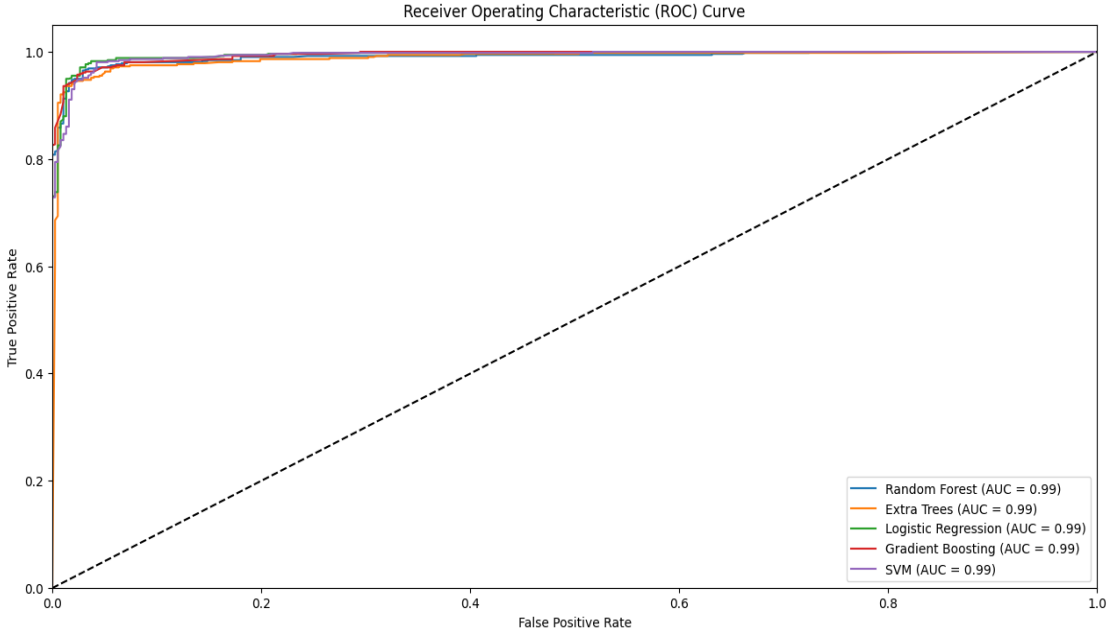

Figure 13: ROC AUC Results of the compared algorithms.

6 Results and discussion of the proposed model and the previous model

Referring to Table 3, the comparison of algorithm accuracies between the developed model and the model in [18], both applied to the same datasets, reveal nuanced performance differences in detecting Android malware. The Logistic Regression model in the developed study demonstrated superior test accuracy of 0.9731, compared to 0.9615 in [18], indicating its enhanced performance on this dataset. Similarly, the Support Vector Machine achieved higher accuracy in the developed model, recording 0.9675 compared to 0.9521 in [18], underscoring its more robust performance on the dataset utilized. The Random Forest and Extra Trees classifiers also demonstrated higher test accuracies in the developed model (Random Forest: 0.9664 to 0.9570; Extra Trees: 0.9608 to 0.9550), suggesting their better adaptation to the dataset's characteristics. In contrast, the Gradient Boosting Classifier showed comparable performance between the developed model (0.9619) and the other model (0.9637), indicating consistent effectiveness across datasets. Overall, the classifiers generally exhibited superior test accuracy in the developed model compared to the results reported in [18]. These observations highlight the importance of dataset selection and evaluation in ensuring machine learning models' robust and reliable performance in practical applications. The proposed Logistic Regression model offers the best accuracy results because it has the highest True Positive and True Negative cases and the lowest False Positive and False Negative Cases. The novelty of the proposed model using machine learning algorithms is provided as follows:

- The dataset contains much data that helps analyze and detect malware on Android platforms.
- The dataset used is rich in features, where these features are taken from Android applications.
- The methodology used can help in detecting Android malware.

The high classification results indicate the high performance of the proposed model using machine learning algorithms.

Table 3: The accuracy results of the developed model and the developed model of [18].

Method	**The Developed Model**	**The Other Model [18]**
Random Forest	0.9664	0.9570
Extra Trees	0.9608	0.9550
Logistic Regression	0.9731	0.9615
Gradient Boosting	0.9619	0.9637
Support Vector Machine	0.9675	0.9521

7 Conclusion and future work

Based on the comprehensive evaluation and analysis presented in this study, the machine learning models, including Random Forest, Extra Trees, Logistic Regression, Gradient Boosting, and Support Vector Machine, have proven highly effective in distinguishing between benign and malware applications within the Android ecosystem. These algorithms demonstrated

robust performance across key metrics such as accuracy, precision, recall, F1-scores, and ROC-AUC, essential for accurate malware detection. Logistic Regression emerged as the standout performer, with the highest overall accuracy of 97.31%, underscoring its reliability in classifying malware instances while maintaining a low false-positive rate.

Comparative analysis with a previous model using the same dataset revealed that our developed model consistently outperformed the prior approach across all evaluated algorithms in accuracy. This improvement signifies the effectiveness of our methodology in enhancing malware detection capabilities on Android devices. The results suggest that our approach achieves superior performance and lays a foundation for future advancements in cybersecurity research.

Moving forward, future research could concentrate on further enhancing the model's scalability and applicability in real-time scenarios. This may entail incorporating more advanced features, exploring ensemble methods to amalgamate different algorithm strengths, or utilizing deep learning techniques to achieve superior detection accuracy. The proposed model can be applied to different datasets to obtain its performance using varying-size datasets. Additionally, broadening the dataset to encompass a more comprehensive array of malware samples and real-world scenarios could offer deeper insights into the model's robustness and applicability.

Furthermore, the continuous evolution of Android malware necessitates ongoing adaptation and innovation in detection methodologies. Tackling these challenges requires interdisciplinary collaboration to develop comprehensive solutions that identify and mitigate the impact of malicious software on mobile device users. This study contributes significantly to safeguarding mobile ecosystems against emerging cybersecurity threats by advancing the forefront of machine learning-driven malware detection. In summary, while this study has made significant strides in enhancing Android malware detection through machine-learning methods, pursuing comprehensive mobile security remains an ongoing endeavor. Embracing innovation and collaboration in future research endeavors can further build upon these foundational advancements to establish more resilient and adaptable defense mechanisms against evolving cyber threats.

Acknowledgment

The authors extend their appreciation to the Arab Open University for funding this work through AOU research fund No. (AOURG-2023-011).

References

[1] 1.Arindaam Roy, Divjeet Singh Jas, Gitanjali Jaggi, and Kapil Sharma, *Android malware detection based on vulnerable feature aggregation.* Procedia Computer Science, 2020. **173**: p. 345-353. DOI: https://doi.org/10.1016/j.procs.2020.06.040.

[2] 2.Hyoil Han, SeungJin Lim, Kyoungwon Suh, Seonghyun Park, Seong-je Cho, and Minkyu Park. *Enhanced android malware detection: An svm-based machine learning approach*. in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*. 2020. IEEE DOI: https://doi.org/10.1109/bigcomp48618.2020.00-96.

[3] 3.V.Maria Anu Sk Heena Kauser, *Android Malware Detection using Machine Learning Techniques KNN-SVM, DBN and GRU*. International Journal of Computer Science and Network Security, 2023. **23**(7): p. 202-209. DOI: https://doi.org/10.22937/IJCSNS.2023.23.7.23.

[4] 4.Chuanchang Liu, Jianyun Lu, Wendi Feng, Enbo Du, Luyang Di, and Zhen Song, *MOBIPCR: Efficient, accurate, and strict ML-based mobile malware detection.* Future Generation Computer Systems, 2023. **144**: p. 140-150. DOI: https://doi.org/10.1016/j.future.2023.02.014.

[5] 5.Vasileios Syrris and Dimitris Geneiatakis, *On machine learning effectiveness for malware detection in Android OS using static analysis data.* Journal of Information Security and Applications, 2021. **59**: p. 102794. DOI: https://doi.org/10.1016/j.jisa.2021.102794.

[6] 6.Halil Murat Ünver and Khaled Bakour, *Android malware detection based on image-based features and machine learning techniques.* SN Applied Sciences, 2020. **2**(7): p. 1299. DOI: https://doi.org/10.1007/s42452-020-3132-2.

[7] 7.Esraa Odat and Qussai M Yaseen, *A novel machine learning approach for android malware detection based on the co-existence of features.* IEEE Access, 2023. **11**: p. 15471-15484. DOI: https://doi.org/10.1109/access.2023.3244656.

[8] 8.Ahmed S Shatnawi, Qussai Yassen, and Abdulrahman Yateem, *An android malware detection approach based on static feature analysis using machine learning algorithms.* Procedia Computer Science, 2022. **201**: p. 653-658. DOI: https://doi.org/10.1016/j.procs.2022.03.086.

[9] 9.GSMA Report, *The Mobile Economy 2024.pd*. 2024.

[10] 10. Muawya Naser, Hussein Albazar, and Hussein Abdel-Jaber, *Mobile Spyware Identification and Categorization: A Systematic Review.* Informatica, 2023. **47**(8). DOI: https://doi.org/10.31449/inf.v47i8.4881.

[11] 11. Maad M Mijwil, *Malware Detection in Android OS Using Machine Learning Techniques.* International Journal of Data Science and Applications, 2020. **3**(2): p. 5-9.

[12] 12. Beenish Urooj, Munam Ali Shah, Carsten Maple, Muhammad Kamran Abbasi, and Sidra Riasat, *Malware detection: a framework for reverse engineered android applications through machine learning algorithms.* IEEE Access, 2022.

10: p. 89031-89050. DOI: https://doi.org/10.1109/access.2022.3149053.

[13] 13. Arvind Mahindru, Himani Arora, Abhinav Kumar, Sachin Kumar Gupta, Shubham Mahajan, Seifedine Kadry, and Jungeun Kim, *PermDroid a framework developed using proposed feature selection approach and machine learning techniques for Android malware detection.* Scientific Reports, 2024. **14**(1): p. 10724. DOI: https://doi.org/10.1038/s41598-024-60982-y.

[14] 14. R Srinivasan, S Karpagam, M Kavitha, and R Kavitha. *An Analysis of Machine Learning-Based Android Malware Detection Approaches*. in *Journal of Physics: Conference Series*. 2022. IOP Publishing DOI: https://doi.org/10.1088/1742-6596/2325/1/012058.

[15] 15. Ali Muzaffar, Hani Ragab Hassen, Michael A Lones, and Hind Zantout, *An in-depth review of machine learning based Android malware detection.* Computers & Security, 2022. **121**: p. 102833. DOI: https://doi.org/10.1016/j.cose.2022.102833.

[16] 16. Naseef-Ur-Rahman Chowdhury, Ahshanul Haque, Hamdy Soliman, Mohammad Sahinur Hossen, Tanjim Fatima, and Imtiaz Ahmed. *Android malware Detection using Machine learning: A Review*. in *Intelligent Systems Conference*. 2023. Springer DOI: https://doi.org/10.36227/techrxiv.22580881.v1.

[17] 17. Jaehyeong Lee, Hyuk Jang, Sungmin Ha, and Yourim Yoon, *Android malware detection using machine learning with feature selection based on the genetic algorithm.* Mathematics, 2021. **9**(21): p. 2813. DOI: https://doi.org/10.3390/math9212813.

[18] 18. BRYAM BLAS RIMAC. *Malware_Prediction_Models_BalancedAccuracy =96.30%*. 2024 [cited 2024 June 15]; Available from: https://www.kaggle.com/code/bryamblasrimac/malware-prediction-models-balancedaccuracy-96-30.

[19] 19. Priya Raghuvanshi and Jyoti Prakash Singh. *Android malware detection using machine learning techniques*. in *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2022. IEEE DOI: https://doi.org/10.1109/csci58124.2022.00200.

[20] 20. Hani AlOmari, Qussai M Yaseen, and Mohammed Azmi Al-Betar, *A comparative analysis of machine learning algorithms for android malware detection.* Procedia Computer Science, 2023. **220**: p. 763-768. DOI: https://doi.org/10.1016/j.procs.2023.03.101.

[21] 21. Ahmed S Shatnawi, Aya Jaradat, Tuqa Bani Yaseen, Eyad Taqieddin, Mahmoud Al-Ayyoub, and Dheya Mustafa, *An android malware detection leveraging machine learning.* Wireless Communications and Mobile Computing, 2022. **2022**(1): p. 1830201. DOI: https://doi.org/10.1155/2022/1830201.

[22] 22. Corentin Rodrigo, Samuel Pierre, Ronald Beaubrun, and Franjieh El Khoury, *BrainShield: a hybrid machine learning-based malware detection model for android devices.* Electronics, 2021. **10**(23): p. 2948. DOI: https://doi.org/10.3390/electronics10232948.

[23] 23. Hanqing Zhang, Senlin Luo, Yifei Zhang, and Limin Pan, *An efficient Android malware detection system based on method-level behavioral semantic analysis.* IEEE Access, 2019. **7**: p. 69246-69256. DOI: https://doi.org/10.1109/access.2019.2919796.

[24] 24. Hayam Alamro, Wafa Mtouaa, Sumayh Aljameel, Ahmed S Salama, Manar Ahmed Hamza, and Aladdin Yahya Othman, *Automated android malware detection using optimal ensemble learning approach for cybersecurity*. IEEE Access, 2023. DOI: http://dx.doi.org/10.1016/j.procs.2020.06.040.

[25] 25. Parnika Bhat and Kamlesh Dutta, *CogramDroid–An approach towards malware detection in Android using opcode ngrams.* Concurrency and Computation: Practice and Experience, 2021. **33**(20): p. e6332. DOI: https://doi.org/10.1002/cpe.6332.

[26] 26. R. Nagarajan S. Muthuselvi, *AN IMPLEMENTATION OF ANDROID MALWARE DETECTION USING MACHINE LEARNING TECHNIQUES.* International Research Journal of Modernization in Engineering Technology and Science, 2024. **06**(03): p. 2543-2548. DOI: https://doi.org/10.56726/irjmets50730.

[27] 27. Suleiman Y Yerima, Sakir Sezer, and Igor Muttik. *Android malware detection using parallel machine learning classifiers*. in *2014 Eighth international conference on next generation mobile apps, services and technologies*. 2014. IEEE DOI: https://doi.org/10.1109/ngmast.2014.23.

[28] 28. Janaka Senanayake, Harsha Kalutarage, and Mhd Omar Al-Kadri, *Android mobile malware detection using machine learning: A systematic review.* Electronics, 2021. **10**(13): p. 1606. DOI: https://doi.org/10.3390/electronics10131606.

[29] 29. Mohammed K Alzaylaee, Suleiman Y Yerima, and Sakir Sezer, *DL-Droid: Deep learning based android malware detection using real devices.* Computers & Security, 2020. **89**: p. 101663. DOI: https://doi.org/10.1016/j.cose.2019.101663.

[30] 30. Jinsung Kim, Younghoon Ban, Eunbyeol Ko, Haehyun Cho, and Jeong Hyun Yi, *MAPAS: a practical deep learning-based android malware detection system.* International Journal of Information Security, 2022. **21**(4): p. 725-738.

DOI: https://doi.org/10.1007/s10207-022-00579-6.

[31] 31. Abdulaziz Alzubaidi, *Sustainable Android Malware Detection Scheme using Deep Learning Algorithm.* International Journal of Advanced Computer Science and Applications, 2021. **12**(12). DOI: https://doi.org/10.14569/ijacsa.2021.01212104.

[32] 32. Muhammad Aamir, Muhammad Waseem Iqbal, Mariam Nosheen, M Usman Ashraf, Ahmad Shaf, Khalid Ali Almarhabi, Ahmed Mohammed Alghamdi, and Adel A Bahaddad, *AMDDLmodel: Android smartphones malware detection using deep learning model.* Plos one, 2024. **19**(1): p. e0296722. DOI: https://doi.org/10.1371/journal.pone.0296722.

[33] 33. Omar N Elayan and Ahmad M Mustafa, *Android malware detection using deep learning.* Procedia Computer Science, 2021. **184**: p. 847-852. DOI: https://doi.org/10.1016/j.procs.2021.03.106.

[34] 34. Ruitao Feng, Sen Chen, Xiaofei Xie, Guozhu Meng, Shang-Wei Lin, and Yang Liu, *A performance-sensitive malware detection system using deep learning on mobile devices.* IEEE Transactions on Information Forensics and Security, 2020. **16**: p. 1563-1578. DOI: https://doi.org/10.1109/tifs.2020.3025436.

[35] 35. Kaggle. *Android Malware Detection Dataset*. 2024 [cited 2024 May 15]; Available from: https://www.kaggle.com/datasets/dannyrevaldo/android-malware-detection-dataset/data.

[36] 36. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg, *Scikit-learn: Machine learning in Python.* the Journal of machine Learning research, 2011. **12**: p. 2825-2830.

[37] 37. Evidently AI. *How to explain the ROC curve and ROC AUC score?* [cited 2024 August 19, 2024]; Available from: https://www.evidentlyai.com/classification-metrics/explain-roc-curve.