UtilitasMathematica

ISSN 0315-3681 Volume 122 (2), 2025

Machine Learning-Powered Dynamic Analysis Framework for Effective Android Malware Detection

[1]Hiral Patel [2]Dr.Mukta Agarwal

[1] Research Scholar, Department of Computer Science, Indus University, Ahmedabad, Gujarat, India
E-mail: [1]patelhiral.21.rs@indusuni.ac.in
[2]Assistant Professor, Department of Computer Science, Indus University, Ahmedabad, Gujarat, India
E-mail: [2]muktaagarwal.dcs@indusuni.ac.in

Abstract

Since Android smartphones are so widely used, hackers have turned them into their main target, which has resulted in a sharp increase in sophisticated mobile malware. Only static analysis techniques are not enough to detect evolving threats due to obfuscation and code transformation techniques. We explore a dynamic analysis-based approach for Android malware detection using machine learning algorithms in this research, leveraging the CICMalDroid 2020 dataset. Various classifiers, including Random Forest, Decision Tree, Naive Bayes, Logistic Regression, AdaBoost, Extra Trees, and Gradient Boosting Machine, were trained using dynamic features that captured runtime behavior, such as system calls, API calls, and other runtime behaviors. An extensive assessment of model performance is made possible by the dataset's inclusion of a wide range of malware families and benign applications. Common metrics like Accuracy, Precision, Recall, F1-score, Specificity, Matthew Correlation Coefficient, Cohen Kappa, and ROC Score were used to evaluate the models. The results of experiments show that machine learning models trained on dynamic features are capable of accurately differentiating between malicious and benign applications, with the highest detection accuracy being attained by the Random Forest classifier.

Keywords: Machine learning, Dynamic analysis, Android malware detection, Feature extraction, Cyber security

1. Introduction

The Android operating system is currently the most widely used mobile platform worldwide. According to a recent report , there are 3.3 billion Android OS users in the world as of 2025 (DemandSage,2025).But greater popularity also means more security risks, particularly malware. Android malware poses a serious threat to user privacy, financial security, and device integrity because it often exploits social engineering, system vulnerabilities, and the spread of malicious apps. Conventional malware detection techniques, like signature-based and rule-based approaches, have become increasingly ineffective against complex and evolving malware variants (Sarma et al., 2012). These methods frequently fail to detect polymorphic malware, which can change their behavior to avoid detection. To get around these limitations, researchers have turned to machine learning (ML) techniques for malware detection. ML-based techniques can learn from historical data and identify risky patterns in apps, even in previously unheard-of samples. Machine learning (ML) has emerged as a potent tool in the cybersecurity arsenal, offering the capability to identify patterns and anomalies indicative of malicious behavior. Static analysis and dynamic analysis are the two main analysis techniques used for feature extraction in machine learning approaches. Static analysis involves examining the application's code and resources without executing it, enabling the detection of known malicious patterns. Static analysis examines the code and manifest files of an application without executing it, Dynamic analysis, on the other hand, observes the application's behavior during execution under controlled conditions, capturing runtime characteristics that may reveal malicious intent (Afonso et al., 2021). However, every strategy has its limitations. Static analysis is often fast, but it can miss runtime features or be evaded by hiding code. On the other hand, dynamic analysis captures behavior during runtime, but it is resource intensive and may not capture all of the code.

Recent studies have demonstrated how well dynamic analysis and machine learning (ML) work together to detect malware. DySign creates behavioral fingerprints of Android apps using dynamic analysis, which makes it easier to identify malware variants using machine learning classification (Karbab, Debbabi, Alrabaee, & Mouheb, 2017). Alzaylaee et al. (2019) presented DL-Droid, a system that outperforms conventional ML-based techniques by using deep learning models trained on dynamic features collected from actual devices. Another study further validated the usefulness of behavioral analysis in Android malware detection by applying Support Vector Regression (SVR) to dynamic features, which produced high classification accuracy and area under the curve (AUC) metrics (Information, 2024).

In this research, we evaluate the use of dynamic features for Android malware classification using the CICMalDroid 2020 dataset—a comprehensive benchmark dataset containing behavioral traces from both benign and malicious applications. The dataset includes multiple malware families, making it suitable for training and evaluating machine learning classifiers under realistic conditions. We experiment with several classical ML algorithms, including Random Forest, Decision Tree, Naive Bayes, Logistic Regression, AdaBoost, Extra Trees, and Gradient Boosting Machine comparing their performance based on metrics such as Accuracy, Precision, Recall, F1-score, Specificity, Matthew Correlation Coefficient, Cohen Kappa, and ROC Score.

The main contributions of this paper are:

- A comprehensive analysis of classical machine learning algorithms on dynamic behavioral data for Android malware detection.
- Utilization of the CICMalDroid 2020 dataset to evaluate detection accuracy across multiple malware families.
- Empirical results demonstrating that dynamic features significantly improve the performance of ML models for Android malware detection.

The remainder of this paper is structured as follows: Section 2 reviews related work,Section 3 details the dataset ,Section 4 outlines the methodology,Section 5 presents the experimental results,and Section 6 concludes with a summary and future work.

2. Related work

Recent studies have explored machine learning approaches for Android malware detection using static analysis. Researchers have employed various algorithms, including Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and other classifiers, to distinguish between benign and malicious applications (Kakavand et al., 2018; Rana et al., 2018). These methods typically analyze features extracted from the Android manifest file and source code, such as permissions, intents, system commands, and API calls (El Fiky et al., 2021). Some studies have combined static and dynamic analysis techniques to improve detection accuracy (Fan Yang et al., 2017). Experiments conducted on datasets like DREBIN and MALGENOME have demonstrated high accuracy rates, with some models achieving over 94% accuracy (Rana et al., 2018; El Fiky et al., 2021).The effectiveness of static analysis and machine learning techniques for detecting Android banking malware, using a dataset of 500 malicious and 500 benign code samples, and finds that the XGboost algorithm achieves high accuracy in malware detection (Ahmed et al., 2023).Propose a machine learning approach for Android malware detection using static features of Android Application Packages (APKs), with the Gaussian Process showing the most promising results in classifying APKs as malware or benign(Alzaylaee, Yerima, & Sezer, 2020).Propose a method for analyzing Android malware using machine learning techniques, specifically static and dynamic analysis, and addresses computational complexity by utilizing a distributed sandbox system based on heterogeneous computing systems with voting methods and ——a rating system(Suresh et al., 2019).Investigate machine learning algorithms for detecting Android malware using a static analysis approach.Collected permissions from each application's APK,generated feature vectors based on extracted permissions and trained several machine learning algorithms to create classification models.Achieving high accuracy with random forest and multi-layer perceptron models(Kumar et al., 2022).A machine learning-based system using static and dynamic analysis with feature selection via PCA and Relief provides an effective method for Android malware detection(Wen, L., & Yu, H. ,2017).Propose a novel method called FWA-FS for Android malware detection using the fireworks algorithm.The method uses static analysis to classify applications as benign or malicious.The strategy enhances classification performance with an average increase of 6% and 25% in accuracy for KNN and Naïve Bayes respectively(Kumar et al., 2022).The Random Forest algorithm achieved the highest accuracy in malware detection with a TPR of 91.6%.Used static analysis focusing on permission-based features from the AndroidManifest.xml file to detect Android malware(Arif et al., 2021).A lightweight Android malware detection system using machine learning with fewer static features is presented.The system uses feature engineering to reduce feature dimensions,making it more efficient.The detection system achieves accuracy and precision above 98% and reduces the feature set size from 3,73,458 to 105 features(Jain & Dave, 2020).A multi-tiered feature selection model is proposed to improve malware detection accuracy by identifying relevant features.The model applies five machine learning

classification techniques to the selected feature set.Random Forest classification achieves the highest accuracy rate of 96.28%(Bhat & Dutta, 2022).Feature selection can reduce the number of Android API calls used for malware detection while maintaining high accuracy using machine learning models(Muzaffar et al., 2022).Propose a novel method of feature selection inspired by TF-IDF and a new method for merging Android application URLs to simplify malware detection.Significantly reduce the feature space and memory size of the final model.The linear support vector machine achieves an accuracy of 99%, which is the highest reported accuracy for the Drebin dataset to date (Salah, Shalabi, & Khedr, 2020).DynaMalDroid, a framework utilizing system calls and various classifiers, achieved up to 99.5% accuracy using Support Vector Machine and AdaBoost (Hashida Haidros Rahima Manzil & Manohar Naik S, 2022). Another study combined static and dynamic analysis, extracting package features, permissions, and dynamic behavior characters, demonstrating improved accuracy over common detection engines (Fan Yang et al., 2017). A comprehensive approach using Androguard and Droidbot for feature extraction, along with supervised learning models, showed that k-nearest neighbor achieved 99% accuracy (Prashant Bhooshan et al., 2024). EnDroid, a dynamic analysis framework, employed multiple behavior features and ensemble learning, with Stacking proving most effective for classification (Pengbin Feng et al., 2018).Presents two machine-learning approaches for detecting and identifying Android malware categories and families.The approaches achieve over 96% accuracy in category detection and over 99% accuracy in family detection.The method provides high-accuracy dynamic analysis while shortening the time required for smartphone malware analysis(El Fiky, Shenawy, & Madkour, 2021).Propose a malware detection system for Android using a combination of static and dynamic analysis with machine learning and deep learning classifiers.The system uses user permissions data for static analysis and network traffic data for dynamic analysis.The multilayer model achieves high accuracy rates for both static and dynamic analysis(Hossain & Riaz, 2021).Propose a hybrid machine learning model efficiently classifies and detects malware in Android apps.The model detects malicious applications quickly and improves Android security compared to existing methods.The model achieved a best accuracy of 100% on benchmark datasets, outperforming state-of-the-art techniques in various metrics(Alam et al., 2023).

3. Dataset

In this study, we use the CICMalDroid 2020 (Mahdavifar et al., 2020) dataset, a comprehensive and publicly available dataset developed by the Canadian Institute for Cybersecurity. This dataset was specifically created for research in Android malware detection, and it contains rich dynamic behavioral data gathered from both benign and malicious Android applications.

CICMalDroid2020, which includes 11,598 most recent samples of five different Android apps categories: Adware, Banking, SMS, Riskware, and Benign. Dataset contains 471 feature columns. Because of its extensive collection of both static and dynamic features, the dataset is appropriate for tasks involving malware detection and classification.

3.1 Data Exploration

To acquire a fundamental understanding of the CICMalDroid 2020 dataset, initial data exploration was carried out. This involved looking at the distribution of class labels (malicious vs. benign), the quantity of samples and features, and locating any missing or unusual values. Patterns and possible problems in the raw data were found using summary statistics and visualizations, which helped direct the feature engineering and cleaning processes that followed. Each row in the dataset represents a single application instance, with features indicating the frequency of specific actions such as ACCESS_PERSONAL_INFO, FS_ACCESS(READ), NETWORK_ACCESS(WRITE), and others. The final column, labeled Class, represents the binary classification target, where 0 denotes benign applications and 1 denotes malicious applications.

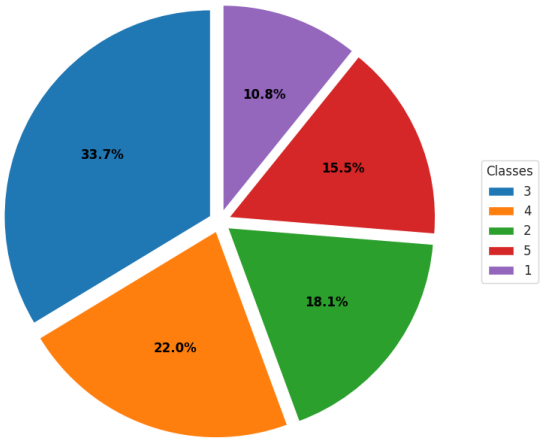

Figure 1 Distribution of classes in CICMalDroid 2020 dataset

Figure-1 shows the pie chart which illustrates the proportional distribution of each class within the CICMalDroid 2020 dataset, including both various malware families and benign applications. This visual representation provides a quick overview of how the dataset is composed in terms of class variety and balance. Dataset contains samples of five different Android apps categories:Adware, Banking, SMS, Riskware, and Benign. Each colored segment represents a class, with its size proportional to the number of samples it contains.Class 1 represents Adware category with 10.8%,Class 2 represents the Banking category with 18.1%,Class 3 represents the SMS category with 33.7%,Class 4 represents the Riskware category with 22.0%,while Class 5 represents the Benign apps with 15.5% of the total samples.

4. Methodology

This study proposes a machine learning-based approach for Android malware classification using dynamic analysis features. The methodology consists of the following key stages:

4.1 Data Preprocessing

Data preprocessing is a crucial step in getting the dataset ready for efficient malware detection using machine learning. We used the CICMalDroid 2020 dataset, which includes dynamic behavioral features recorded while Android apps were running, in this investigation. The objective was to transform these unprocessed features into a clean, normalized dataset that could be used to train trustworthy classification models.

4.1.1 Data Cleaning

We thoroughly analyze missing and zero values across all features in order to guarantee data quality and integrity.Missing values in numerical behavior-based features were replaced with 0.0 under the assumption that missingness indicates absence of that activity. If a column had no discriminatory power, it was considered for removal if it had consistently missing or zero values throughout the dataset.Samples with missing critical labels or values were eliminated for features (e.g., class imbalance) where missing data could affect model results. To visually assess the presence and distribution of missing values, a heatmap was generated using Seaborn's heatmap() function. Each cell in the heatmap corresponds to a feature value, with missing entries highlighted distinctly. This graphical representation enabled a quick, intuitive understanding of missing data patterns across all features. It was observed that only a few columns exhibited sparsely missing values, which were either imputed or removed based on their significance to the classification task. Figure-2 shows the heatmap visualisation of missing values across features in the CICMalDroid 2020 dataset.

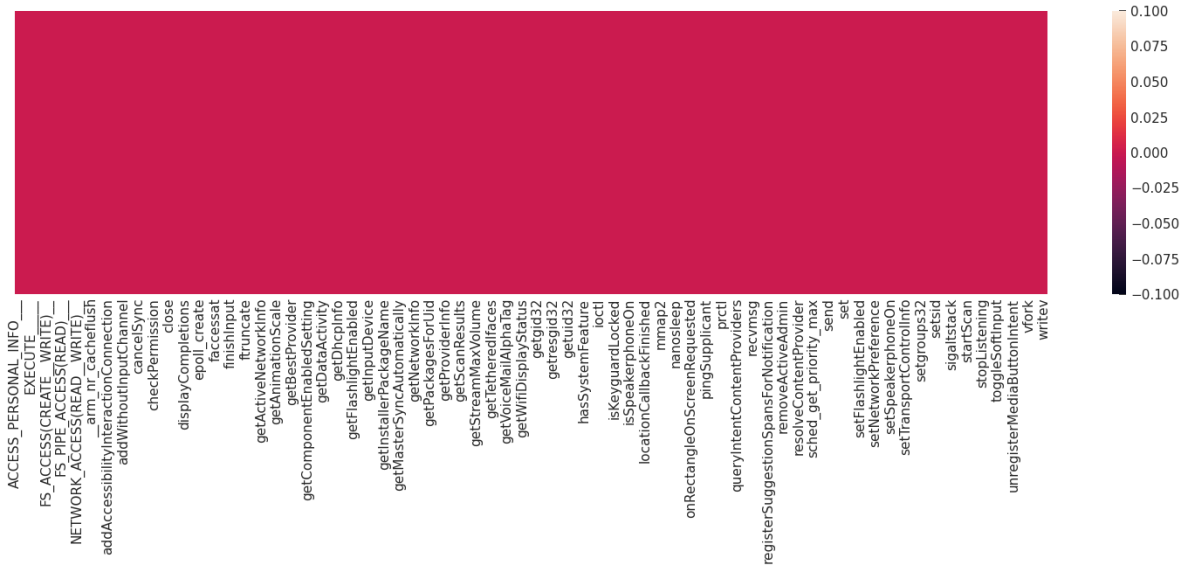

Figure 2 Heatmap visualisation missing values across features in the CICMalDroid 2020 dataset

4.2 Correlation Analysis

To better understand the relationships among the numeric features and improve the quality of input data, a correlation matrix was computed for all continuous features in the dataset. The correlation matrix measures the degree to which variables are linearly related, with values ranging from -1 to +1, where values close to +1 or -1 indicate strong positive or negative linear relationships, respectively, and values near 0 suggest weak or no linear correlation.

- Highly correlated features introduce redundancy and multicollinearity, which can negatively affect the performance of certain models, especially linear classifiers. Identifying such features enables their removal or transformation, thereby stabilizing the model and improving interpretability.
- Redundant features that convey overlapping information can be eliminated, reducing the dimensionality of the dataset. This process improves computational efficiency and mitigates overfitting.
- Features with significant correlation to the class label (e.g., benign or malicious) are likely to be more informative for classification tasks. These were prioritized during model training. Figure 3 shows heatmap visualization of the correlation matrix that shows the pairwise linear relationships among numeric dynamic features.

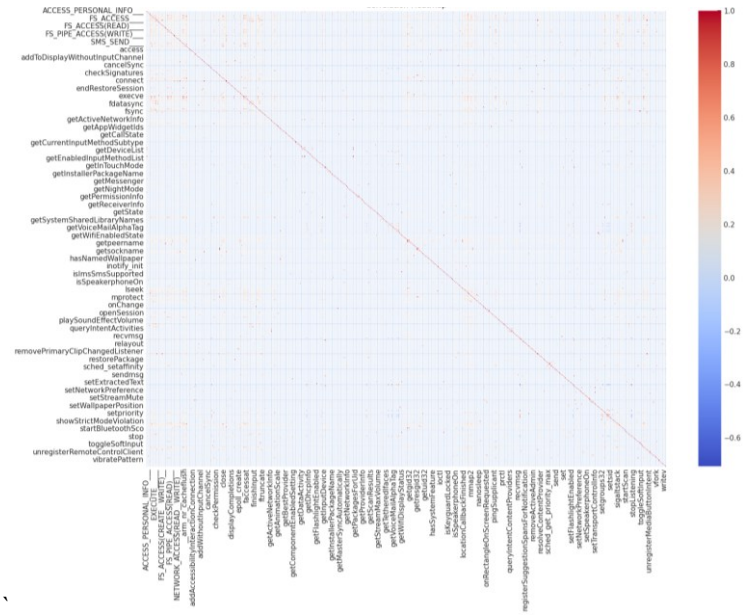

Figure 3 Heatmap visualization of the correlation matrix showing the pairwise linear relationships among numeric dynamic features

4.3 Feature Selection

To enhance the effectiveness and interpretability of the machine learning models, a feature selection process was conducted using a Random Forest classifier. This method evaluates the importance of each feature based on how much it improves the model's prediction accuracy across multiple decision trees. The approach is both model-driven and data-informed, offering a reliable means of identifying which features contribute most significantly to malware classification.

After training the Random Forest classifier on the full feature set, feature importance scores were extracted and ranked. The top-ranked features represent dynamic behavioral characteristics of android applications that are most indicative of malicious activity. These features include system calls, file access operations, and device information queries, which are commonly exploited by malware to perform stealthy or unauthorized actions.

As shown in figure 4, features such as pread64, access, and stat64 had the highest importance scores. These features are indicative of file handling and system access behavior, which are often manipulated by malicious applications. Functions like getDeviceId, ACCESS_PERSONAL_INFO__, and getDisplayInfo suggest access to sensitive user or device data—an activity typically associated with spyware and other intrusive malware types.By focusing on the most informative features, the dimensionality of the data is effectively reduced, helping to mitigate overfitting and improve model training efficiency. Moreover, retaining only the most relevant features simplifies model interpretation, enabling security analysts to better understand and trust the model's decisions.

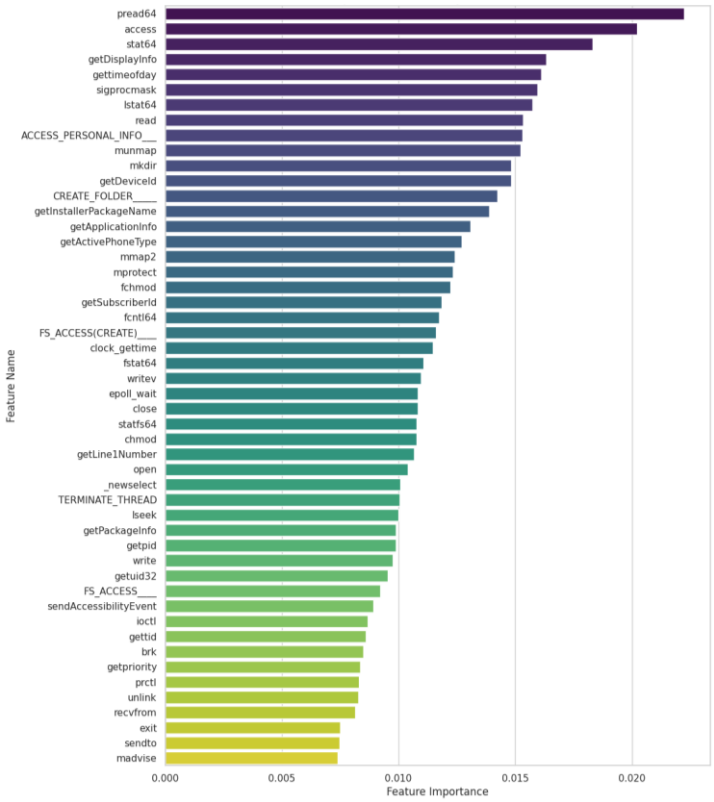

Figure 4 Top most important features selected by Random Forest

In this study, the pairplot was applied to a selected subset of dynamic behavioral features such as ACCESS_PERSONAL_INFO___, CREATE_PROCESS_____, EXECUTE_____, FS_ACCESS____, and write. These features were previously identified through Random Forest-based feature importance analysis as being highly indicative of malicious behavior. The visualization revealed that certain combinations of these features showed visible clustering and separation between the two classes, suggesting that they possess strong discriminatory power.Using Seaborn's pairplot, pairwise relationships between the selected features were plotted. Each data point in the scatter plots is color-coded according to its class label—either benign or

Class 1 - Adware Class 2-Banking Class 3 - SMS Class 4 -Riskware Class 5- Benign

Figure 5 Pairplot for Selected Feature

malicious enabling visual inspection of class separability. Transparency (alpha=0.7) was applied to better observe overlapping points, and the layout was optimized for clarity and readability.

4.4 Feature Scaling

To ensure that all numeric features contributed equally to the learning algorithms, feature scaling was applied as part of the machine learning pipeline. This step was conditionally executed based on a configurable flag in the implementation, allowing for controlled experimentation with and without scaling. Standardization was performed using the StandardScaler from the Scikit-learn library (Pedregosa et al., 2011), which transforms each feature by subtracting the mean and dividing by the standard deviation.

$$x' = \frac{x-\mu}{\sigma} \qquad (1)$$

where x is the original feature value, μ is the mean, and σ is the standard deviation of the feature in the training dataset.The training data was scaled using the fit_transform() method, which computes the parameters μ and σ, while the test data was transformed using the same parameters via the transform() method to maintain consistency and prevent data leakage. For algorithms that depend on distance or margin calculations, feature scaling is especially crucial because these algorithms are sensitive to the magnitude of input features. Larger numerical range features could disproportionately affect the model without scaling, resulting in less-than-ideal performance. By transforming all features to a standardized scale, the training process becomes more stable, improves convergence speed, and enhances the generalization ability of the models.

5. Experiment and Results

The experiment was conducted using the CICMalDroid 2020 dataset, which comprises dynamic behavioral features extracted from Android applications. We used 11,598 APK samples from the dataset CICMalDroid 2020.Then a dynamic analysis was carried out to capture system calls,file access operations and device

information queries made by applications. Using feature engineering chooses relevant features for the training set by employing the feature selection technique. Finally, several machine learning algorithms are utilized in the detection module to identify whether the application is malware.Several machine learning techniques including Random Forest, Decision Tree, Naive Bayes, Logistic Regression, AdaBoost, Extra Trees, and Gradient Boosting Machine are employed in this final detection module of the proposed framework. The sample data, chosen features, and application scenario are frequently taken into account while choosing a model algorithm. In the training phase, the dataset is trained with the selected features which are extracted from the feature engineering. During testing, the classifier's outputs are evaluated using the performance metrics.

Table 1 Experiment Results

classifier	Accuracy	Precision	Recall	F1 Score	Specificity	Matthew Correlation Coefficient	Cohen Kappa	ROC Score
Naive Bayes	0.569397	0.355851	0.211374	0.245397	0.943396	0.446854	0.408847	0.814259
Logistic Regression	0.862931	0.550634	0.508367	0.52864	0.879227	0.821887	0.819944	0.945269
Decision Tree	0.905172	0.558886	0.57098	0.564282	0.955752	0.876943	0.876778	0.920114
Random Forest	0.946121	0.613295	0.610757	0.611306	0.995745	0.930268	0.930047	0.995292
AdaBoost	0.826293	0.568026	0.432293	0.490683	0.987179	0.775358	0.770098	0.961611
Extra Trees	0.942672	0.613580	0.612619	0.612271	0.995781	0.9259	0.925601	0.994162
Gradient Boosting Machine	0.92931	0.602041	0.595096	0.598235	0.99115	0.908204	0.908105	0.991291

Table-1 displays the results that were obtained in the form of metrics such as Accuracy, Precision, Recall, F1-score, Specificity, Matthew Correlation Coefficient, Cohen Kappa, and ROC Score. The corresponding graphical representation of experimental results is shown in figure-6.

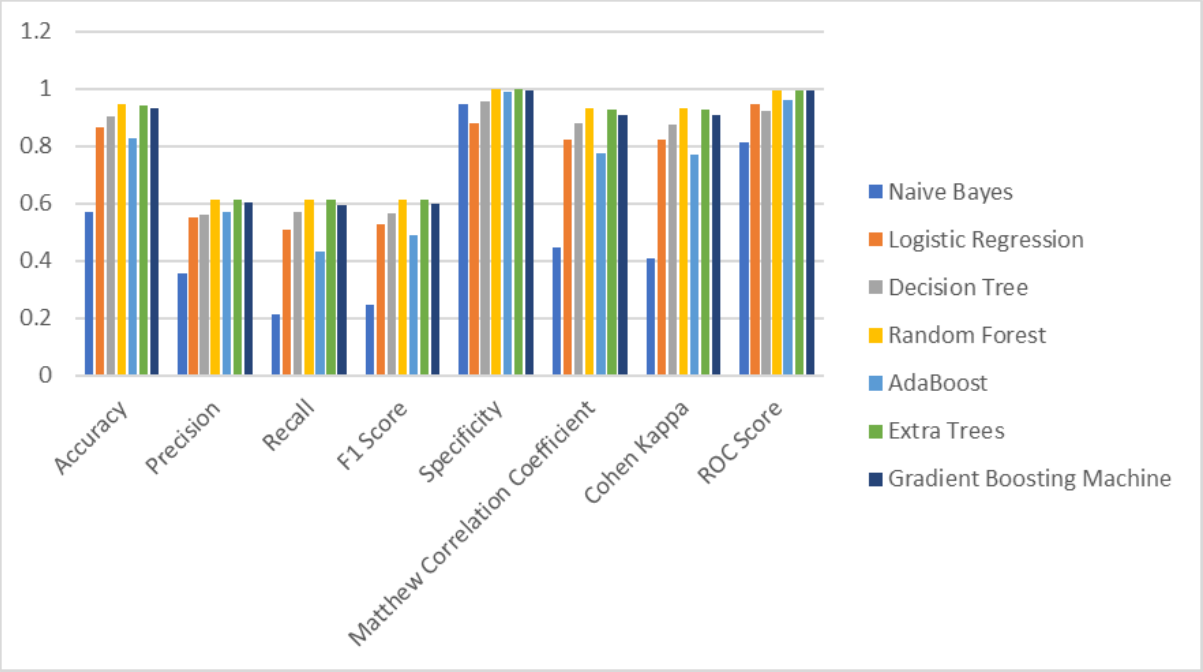

Figure 6 Performance of different classifiers

The results show that the Random Forest classifier had the highest accuracy of 94 % and Naive Bayes had the lowest accuracy of 56%. The confusion matrices of results with Naive Bayes and Random Forest techniques are shown in Figure 7 and Figure 8 respectively.

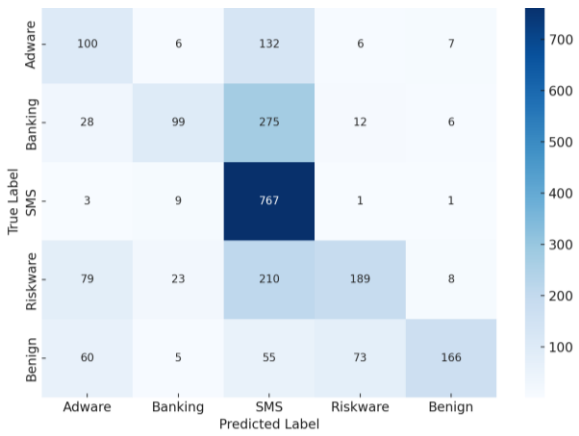

Figure 7 Confusion Matrix -Naive Bayes Classifier

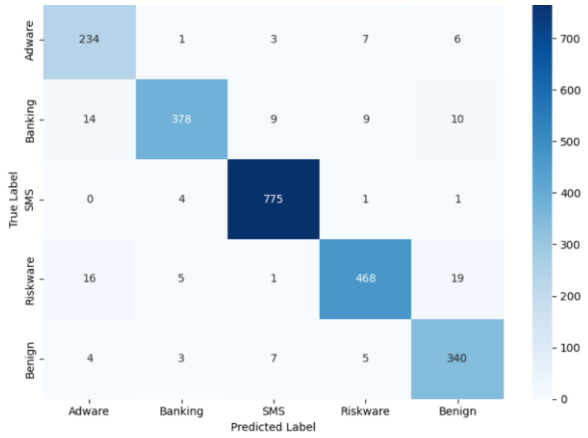

Figure 8 Confusion Matrix - Random Forest Classifier

6.Conclusion

Innovation in technology has increased consumers' reliance on mobile applications. Because of its popularity and affordability, Android has become the preferred choice for the majority of customers. This incentivizes cybercriminals to develop and distribute malicious applications through the official Google Play Store and other unaffiliated sources. As a result, this subject requires extensive research studies. In this paper, we present a dynamic analysis-based approach to detect malicious applications in Android devices using Machine learning classifiers. Using the CICMalDroid 2020 dataset, various classifiers were evaluated, including Random Forest, Decision Tree, Naive Bayes, Logistic Regression, AdaBoost, Extra Trees, and Gradient Boosting Machine.. Among these, Random Forest achieved the highest performance with an accuracy of 94.61%, followed closely by Extra Trees, and Gradient Boosting Machine.The experimental results, visualized through bar charts and confusion matrices, affirm that dynamic analysis significantly enhances detection accuracy. The confusion matrix revealed that the classifiers were particularly effective at distinguishing benign apps from various malware families such as Adware, Banking Trojans, SMS malware, and Riskware.

Overall, the study underscores the importance of using comprehensive features and ensemble models to improve detection rates and reduce false positives. Future work could explore deep learning approaches and real-time detection systems, further strengthening Android security in an increasingly complex threat landscape.

References

1. Afonso, V. M., Vieira, F., Canuto, A. M., & Silva, A. R. (2021). A hybrid approach for detecting Android malware using machine learning. *Electronics*, *10*(13), 1606. https://doi.org/10.3390/electronics10131606
2. DemandSage (2025). *Android statistics: Market share, users & more*. Retrieved February 19, 2025, from https://www.demandsage.com/android-statistics/
3. Sarma, B., Li, N., & Bobba, R. (2012). Android permissions: A perspective combining risks and benefits. *IET Information Security*, *6*(3), 113–120. https://doi.org/10.1049/iet-ifs.2011.0123
4. Karbab, E. B., Debbabi, M., Alrabaee, S., & Mouheb, D. (2017). DySign: Dynamic fingerprinting for the automatic detection of Android malware. *arXiv preprint* arXiv:1702.05699. https://arxiv.org/abs/1702.05699
5. Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2019). DL-Droid: Deep learning-based Android malware detection using real devices. *Computers & Security, 89*, 101663. https://doi.org/10.1016/j.cose.2019.101663
6. Information. (2024). Android malware detection using support vector regression for dynamic feature analysis. *Information, 15*(10), 658. https://doi.org/10.3390/info15100658
7. Mahdavifar, S., Abdul Kadir, A. F., Fatemi, R., Alhadidi, D., & Ghorbani, A. A. (2020). Dynamic Android malware category classification using semi-supervised deep learning. *Proceedings of the 18th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, 17–24. https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00021
8. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

9. Yang, F., Zhuang, Y., & Wang, J. (2017). Android malware detection using hybrid analysis and machine learning technique. In *Cloud Computing and Security: Third International Conference, ICCCS 2017, Nanjing, China, June 16-18, 2017, Revised Selected Papers, Part II 3* (pp. 565-575). Springer International Publishing. https://doi.org/10.1007/978-3-319-68542-7_48
10. Kakavand, M., Dabbagh, M., & Dehghantanha, A. (2018, November). Application of machine learning algorithms for android malware detection. In *Proceedings of the 2018 International Conference on Computational Intelligence and Intelligent Systems* (pp. 32-36). https://doi.org/10.1145/3293475.3293489
11. El Fiky, A. H., Elshenawy, A., & Madkour, M. A. (2021, May). Detection of android malware using machine learning. In *2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)* (pp. 9-16). IEEE. https://doi.org/10.1109/MIUCC52538.2021.9447661
12. Ahmed, A. M., Saeed, M. A., Hamood, A. A., Alazab, A. A., & Ahmed, K. A. (2023, October). Comparative Study of Static Analysis and Machine Learning Approaches for Detecting Android Banking Malware. In *2023 3rd International Conference on Emerging Smart Technologies and Applications (eSmarTA)* (pp. 01-08). IEEE. https://doi.org/10.1109/eSmarTA59349.2023.10293602
13. Al Zaabi, A., & Mouheb, D. (2020, November). Android malware detection using static features and machine learning. In *2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)* (pp. 1-5). IEEE. https://doi.org/10.1109/CCCI49893.2020.9256450
14. Raghuraman, C., Suresh, S., Shivshankar, S., & Chapaneri, R. (2020). Static and dynamic malware analysis using machine learning. In *First International Conference on Sustainable Technologies for Computational Intelligence: Proceedings of ICTSCI 2019* (pp. 793-806). Springer Singapore.https://doi.org/10.1007/978-981-15-0029-9_62
15. Guendouz, M., & Amine, A. (2022). A Comparative Study of Machine Learning Techniques for Android Malware Detection. *International Journal of Software Innovation (IJSI)*, *10*(1), 1-13. https://doi.org/10.4018/ijsi.309719
16. Wen, L., & Yu, H. (2017). An Android malware detection system based on machine learning. *AIP Conference Proceedings, 1864*(1), 020136. https://doi.org/10.1063/1.4992953
17. Guendouz, M., & Amine, A. (2022). A New Wrapper-Based Feature Selection Technique with Fireworks Algorithm for Android Malware Detection. *Int. J. Softw. Sci. Comput. Intell., 14*, 1-19. https://doi.org/10.4018/ijssci.312554
18. Arif, J. M., Ab Razak, M. F., Awang, S., Tuan Mat, S. R., Ismail, N. N., & Firdaus, A. (2021). A static analysis approach for Android permission-based malware detection systems. *PLoS ONE, 16*(3), e0248586. https://doi.org/10.1371/journal.pone.0248586
19. Jain, K., & Dave, M. (2020). Machine Learning-Based Lightweight Android Malware Detection System with Static Features.https://doi.org/10.1007/978-981-15-7804-5_26
20. Bhat, P., & Dutta, K. (2021). A multi-tiered feature selection model for android malware detection based on Feature discrimination and Information Gain. *J. King Saud Univ. Comput. Inf. Sci., 34*, 9464-9477.https://doi.org/10.1016/j.jksuci.2021.11.004
21. Muzaffar, A., Ragab Hassen, H., Lones, M.A., & Zantout, H. (2022). Android Malware Detection Using API Calls: A Comparison of Feature Selection and Machine Learning Models. *Lecture Notes in Networks and Systems*. https://doi.org/10.1007/978-3-030-95918-0_1
22. Salah, A., Shalabi, E., & Khedr, W. I. (2020). A lightweight Android malware classifier using novel feature selection methods. Symmetry, 12(6), 945. https://doi.org/10.3390/sym12060945
23. Haidros Rahima Manzil, H., & Naik S, M. (2022). DynaMalDroid: Dynamic Analysis-Based Detection Framework for Android Malware Using Machine Learning Techniques. *2022 International Conference on Knowledge Engineering and Communication Systems (ICKES)*, 1-6.https://doi.org/10.1109/ICKECS56523.2022.10060106
24. Yang, F., Zhuang, Y., & Wang, J. (2017). Android Malware Detection Using Hybrid Analysis and Machine Learning Technique. *International Conference on Communication, Computing & Security*.https://doi.org/10.1007/978-3-319-68542-7_48
25. Bhooshan, P., L, S.D., & Sonkar, N. (2024). Comprehensive Android Malware Detection: Leveraging Machine Learning and Sandboxing Techniques Through Static and Dynamic Analysis. *2024 IEEE 21st International Conference on Mobile Ad-Hoc and Smart Systems (MASS)*, 580-585. https://doi.org/10.1109/MASS62177.2024.00092
26. Feng, P., Ma, J., Sun, C., Xu, X., & Ma, Y. (2018). A Novel Dynamic Android Malware Detection System With Ensemble Learning. *IEEE Access, 6*, 30996-31011. https://doi.org/10.1109/ACCESS.2018.2844349
27. Fiky, A.H., Shenawy, A.E., & Madkour, M.A. (2021). Android Malware Category and Family Detection and Identification using Machine Learning. *ArXiv, abs/2107.01927*.
28. Hossain, M.S., & Riaz, M.H. (2021). Android Malware Detection System: A Machine Learning and Deep Learning Based Multilayered Approach. *ICO*. https://doi.org/10.1007/978-3-030-93247-3_28
29. Bashir, S., Hussain, F.M., Khan, F.H., & Abid, A.S. (2023). Hybrid machine learning model for malware analysis in android apps. *Pervasive Mob. Comput., 97*, 101859. https://doi.org/10.1016/j.pmcj.2023.101859