Journal of Academic Research (Applied Sciences), Vol 30, Issu 01, 2026 "43-51"

A Stacking Framework for Malware Detection with XGBoost and Random Forest

Mohammed M Elsheh[1] **Fatima A Aboudabous**[1]

1- Department of Computer Science, Libyan Academy- Misrata, Libya

m.elsheh@lam.edu.ly fatima.aboudabous@it.lam.edu.ly

Article information

Key words

XGBoost, Feature Selection, Stacking Ensemble, CICMalDroid 2020, SMOTE, Malware Detection

Received 07 01 2026, Accepted 23 01 2026, Available online 24 01 2026

Abstract

Short abstract. Android devices are increasingly targeted by malware due to their widespread adoption and open ecosystem. Traditional signature-based detection methods are insufficient for evolving threats, highlighting the need for intelligent approaches. This study proposes an enhanced Android malware classification framework using XGBoost, Random Forest for feature selection, and a stacking ensemble. The CICMalDroid 2020 dataset was preprocessed through outlier removal, robust scaling, binary transformation, and SMOTE-based class balancing. Experimental results show that the stacking ensemble achieves the highest detection performance with **98.14% accuracy**, **93.62% F1-score**, and **99.63% ROC-AUC**, outperforming individual models. Additionally, using RF as features selector improves computational efficiency, reducing training and testing times. These findings demonstrate that integrating advanced machine learning techniques enhances both the effectiveness and efficiency of Android malware detection.

I. Introduction

Android is currently the most widely adopted mobile operating system worldwide, which has significantly increased its exposure to various security threats. The open architecture of Android and the availability of third-party application markets have made it an attractive target for malware developers. Android malware aims to compromise user privacy, steal sensitive information, and cause financial or operational damage, posing serious risks to both individuals and organizations [1].

Android malware manifests in diverse forms such as trojans, spyware, ransomware, and adware, with continuously evolving behaviors designed to evade traditional security mechanisms. Early studies revealed that signature-based detection techniques are insufficient to handle the rapid evolution and obfuscation strategies of modern malware, emphasizing the need for more intelligent and adaptive solutions [1].

In recent years, machine learning (ML) techniques have become a dominant approach for Android malware detection due to their ability to learn complex patterns from application features and behaviors. Numerous studies have demonstrated that ML-based classifiers outperform traditional rule-based methods, particularly when trained on large-scale datasets [2],

m.elsheh@lam.edu.ly

[3]. However, the effectiveness of these models is often challenged by high-dimensional feature spaces and imbalanced datasets, which can degrade classification performance.

To address these challenges, researchers have increasingly adopted ensemble learning techniques, which combine multiple classifiers to improve robustness and generalization. Stacking ensemble frameworks, in particular, have shown promising results by leveraging the complementary strengths of different models [3]. In parallel, feature selection methods have been employed to reduce redundancy, improve computational efficiency, and enhance detection accuracy [4], [5].

Motivated by these observations, this study proposes an optimized Android malware detection framework that integrates feature selection with stacking ensemble learning. By combining efficient feature reduction techniques with advanced ensemble classifiers, the proposed approach aims to improve both detection performance and computational efficiency, addressing key limitations identified in recent studies.

II. Related Work

Early research on Android malware primarily aimed at understanding its core characteristics and tracking its evolution over time. Zhou and Jiang [1] conducted one of the first comprehensive analyses of Android malware, focusing on the behavioral patterns of malicious applications. By applying both static and dynamic analysis to a large collection of real-world malware samples, their study revealed the limitations of traditional signature-based detection systems. Although machine learning was not employed, their work laid a critical foundation for subsequent research that leveraged ML techniques for malware detection.

With the growing availability of Android datasets, recent studies have increasingly adopted machine learning techniques to improve detection accuracy. Raval and Anwar [2] studied Android malware detection using machine learning, comparing classifiers such as Random Forest, MLP, Logistic Regression, XGBoost, SVM, and AdaBoost. They found that tree-based and ensemble models, like Random Forest and XGBoost, generally perform better than simpler methods, achieving high accuracy and AUC. The study also highlighted the importance of feature selection and preprocessing, while noting challenges such as high-dimensional data and class imbalance.

Feature engineering and feature selection have been widely explored to address these challenges. Saleem et al. [3] investigated the effectiveness of permission-based feature ranking methods to reduce feature dimensionality and improve detection efficiency. Using Android application datasets, they evaluated several machine learning classifiers, including Random Forest, Decision Tree, and SVM, combined with Chi-square and Fisher's Exact Test for feature selection. Their experimental results demonstrated that Random Forest coupled with optimized permission ranking achieves very high detection accuracy and F1-scores, confirming the importance of selecting discriminative features in Android malware detection systems.

Building on this direction, Ansori et al. [4] proposed an Android malware classification framework that integrates gain ratio-based feature selection with ensemble learning. Their study addressed the problem of redundant and irrelevant features degrading classifier performance. Using the CICMalDroid 2020 dataset, they evaluated multiple ensemble and individual classifiers. The results indicated that gain ratio feature selection consistently improved classification accuracy and computational efficiency across different models, reinforcing the role of feature selection in large-scale Android malware analysis.

In parallel, explainable machine learning has gained attention to address the lack of transparency in malware detection systems. Palma et al. [5] focused on developing explainable ML models for Android malware detection, aiming to identify the most influential features contributing to classification decisions. Using public Android datasets, they applied machine learning classifiers alongside explainability techniques to analyze feature importance. Their findings showed that permission-based features play a critical role in detection accuracy, while explainable models help improve trust and interpretability without significantly compromising performance.

More advanced ensemble strategies emerged to improve robustness and generalization. Wang et al. [6] introduced MFDroid, a stacking ensemble framework combining diverse base learners to address challenges such as obfuscation and class imbalance. Tested on multiple Android malware datasets, including CICMalDroid variants, MFDroid consistently outperformed individual classifiers in detection accuracy and stability.

Recent work has integrated explainability within ensemble frameworks. Adriansyah et al. [7] proposed a model that combined feature selection, ensemble learning, and SHAP-based explainable AI, achieving both high classification performance and interpretability on Android malware datasets. Similarly, Alsharif and Alharby [8] demonstrated that ensemble methods could effectively handle multi-class malware classification, surpassing single classifiers across various evaluation metrics.

Most recently, Alhogail and Alharbi [9] evaluated several traditional ML classifiers, including Random Forest and SVM, combined with feature selection on the CICMalDroid 2020 dataset. Their results showed that Random Forest achieved superior detection and categorization performance. However, their framework primarily relied on individual classifiers, highlighting an opportunity for further improvements using optimized stacking ensembles to enhance robustness and computational efficiency.

Despite these advances, existing studies tend to focus on detection accuracy while treating feature selection, ensemble learning, and computational efficiency separately. Few have systematically integrated Random Forest-based feature selection with modern gradient boosting models, such as XGBoost, within a fully optimized stacking ensemble framework. Addressing this gap, the present study proposes an efficient stacking-based Android malware detection framework, evaluated on the CICMalDroid 2020 dataset, aiming to combine high detection accuracy, robust generalization, and computational efficiency.

III. Research Design

The proposed framework is constructed of several phases as illustrated in Fig 1. Three models were trained: baseline XGBoost**,** XGBoost with selected features**,** and a stacking ensemble combining multiple classifiers with XGBoost as a meta-learner. All models were evaluated on a test set using accuracy, precision, recall, F1-score, ROC-AUC**,** and execution time is computed to compare detection performance and computational efficiency.

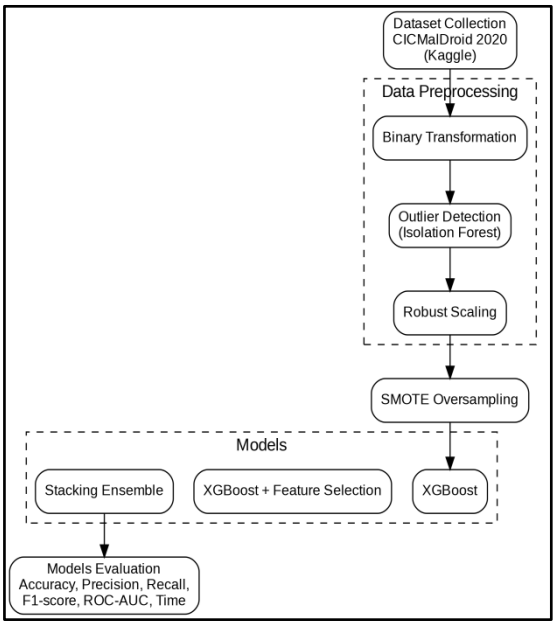

Figure 1. Overview of the proposed malware detection framework

A. Data Description

This study utilizes the CICMalDroid 2020 dataset, which was obtained from the Kaggle platform. The dataset consists of 11,598 Android applications, including both malicious and benign samples. The malware samples are categorized into four classes: Adware (1,253 samples)**,** Banking malware (2,100 samples)**,** SMS malware (3,904 samples)**,** and Riskware (2,546 samples), along with 1,795 benign applications. This diverse distribution makes the dataset suitable for evaluating Android malware classification models.

B. Data preprocessing

Data preprocessing was applied to the CICMalDroid 2020 dataset to improve data quality and enhance classification performance. The preprocessing pipeline included outlier detection, feature scaling, binary transformation, and handling class imbalance.

- **Outlier Detection:** Three outlier detection methods were evaluated: Z-score, Interquartile Range (IQR), and Isolation Forest. Z-score and IQR significantly reduced the dataset size, indicating overly aggressive filtering in high-dimensional malware data. In contrast, Isolation Forest removed only a small portion of noisy samples, reducing the dataset from 11,598 to 11,022 samples while preserving most of the data. Due to its robustness and minimal information loss, Isolation Forest was selected for subsequent experiments.
- **Feature Scaling:** To ensure that all numerical features contribute equally to the learning process and to reduce the influence of extreme values, Robust Scaling was applied. This method, based on the median and IQR, is less sensitive to outliers and is well-suited for Android malware features. After scaling, the dataset retained 11,022 samples with 472 normalized features.
- **Binary Transformation:** The original five-class dataset was converted into a binary classification problem to focus on malware detection. All malware categories were merged into a single **Malware** class (1), while benign applications were labeled as

Benign (0). This simplification enables clearer evaluation and more effective learning for security-oriented classification tasks.

C. Data Balancing

The binary dataset exhibited significant class imbalance, with malware samples greatly outnumbering benign ones. To address this issue, the Synthetic Minority Over-Sampling Technique (SMOTE) was applied to the training set. SMOTE generated synthetic benign samples, resulting in a balanced class distribution and reducing classifier bias toward the majority class. This approach improves detection performance, particularly in terms of recall and F1-score**.**

D. Modeles Architectures

This study investigates three different ML configurations for Android malware detection using the CICMalDroid 2020 dataset: a baseline XGBoost model, XGBoost combined with feature selection, and a stacking ensemble model. Each configuration is designed to progressively enhance learning capability, efficiency, and generalization.

- **XGBoost Model:** XGBoost was employed as the primary classification model due to its strong performance in high-dimensional and imbalanced datasets. It is an ensemble gradient boosting method that builds decision trees sequentially, where each new tree focuses on correcting the errors of previous ones. XGBoost incorporates regularization techniques to reduce overfitting, supports parallel computation for scalability, and can handle missing values automatically, making it suitable for large-scale Android malware detection tasks.
- **XGBoost with Feature Selection:** To reduce dimensionality and improve computational efficiency, feature selection was integrated prior to training XGBoost. Random Forest feature importance was used to identify the most informative features related to malware behavior, particularly those associated with API calls and file system activities. By training XGBoost on a reduced subset of features, this configuration aims to minimize noise, enhance interpretability, and accelerate training while preserving discriminative information.
- **Stacking Ensemble Model:** A stacking ensemble approach was implemented to further improve classification robustness. The ensemble consists of two levels: Level-1 includes diverse base learners (XGBoost, LightGBM, Random Forest, and Logistic Regression) to capture different linear and non-linear patterns in the data. Level-2 employs XGBoost as a meta-learner to combine the predictions of the base models optimally. This architecture leverages model diversity to improve generalization and is particularly effective for complex, high-dimensional malware datasets.

E. Evaluations Metrics

The evaluation of the proposed models was conducted using standard metrics commonly applied in malware detection studies, including accuracy**,** precision**,** recall**,** F1-score**,** and ROC-AUC**.** Accuracy measures the proportion of correctly classified applications among all samples. Precision reflects the proportion of correctly detected malware instances among all samples classified as malware. Recall indicates the model's ability to identify all actual malware instances. F1-score provides a balanced measure combining precision and recall, which is particularly important for imbalanced datasets. ROC-AUC assesses the model's

discriminative capability between malware and benign classes across varying thresholds, offering a comprehensive view of classification performance.

IV. Results and Discussion

To evaluate the proposed Android malware detection framework, three models were assessed: XGBoost**,** XGBoost with feature selection**,** and a stacking ensemble model. The experiments were conducted on the preprocessed CICMalDroid 2020 dataset, which was divided into training and testing sets.

The baseline XGBoost model achieved an accuracy of 97.73%**,** an F1-score of 92.28%**,** and a ROC-AUC of 99.66**%**, demonstrating strong malware detection capability. Incorporating feature selection led to a slight performance improvement, increasing accuracy to 97**.**78% and recall to 93.21%**,** indicating better sensitivity to malicious applications.

The stacking ensemble model achieved the best overall performance, with an accuracy of 98.14%**,** precision of 94.36%**,** recall of 92.90%**,** an F1-score of 93.62%**,** and a ROC-AUC of 99.63%**.** These results confirm that combining multiple learning models improves detection effectiveness. Table 1 summarizes the comparative performance of the evaluated models.

Table.1 comparative performance of the three models

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
XGBoost	97.73%	92.28%	92.28%	92.28%	99.66%
XGBoost with Feature Selection	97.78%	91.79%	93.21%	92.50%	99.63%
stacking ensemble	98.14%	94.36 %	92.90%	93.62%	99.63%

As shown in Fig. 2, In addition to classification performance, the execution time of the evaluated models was analyzed to assess their efficiency. The baseline XGBoost model required 12.9 **s** for training and 0.054 **s** for testing. When feature selection was applied, the training time decreased significantly to **7.**45 **s**, and the testing time was reduced to 0.036 **s**, confirming that dimensionality reduction improves computational efficiency without degrading performance.

In contrast, the stacking ensemble model exhibited substantially higher execution times, requiring 102.06 **s** for training and 21.49 **s** for testing. This increase is attributed to the complexity of combining multiple base learners and a meta-learner. Although the ensemble achieves superior detection accuracy, its higher computational cost makes it more suitable for offline or batch-based malware analysis scenarios rather than time-sensitive applications.

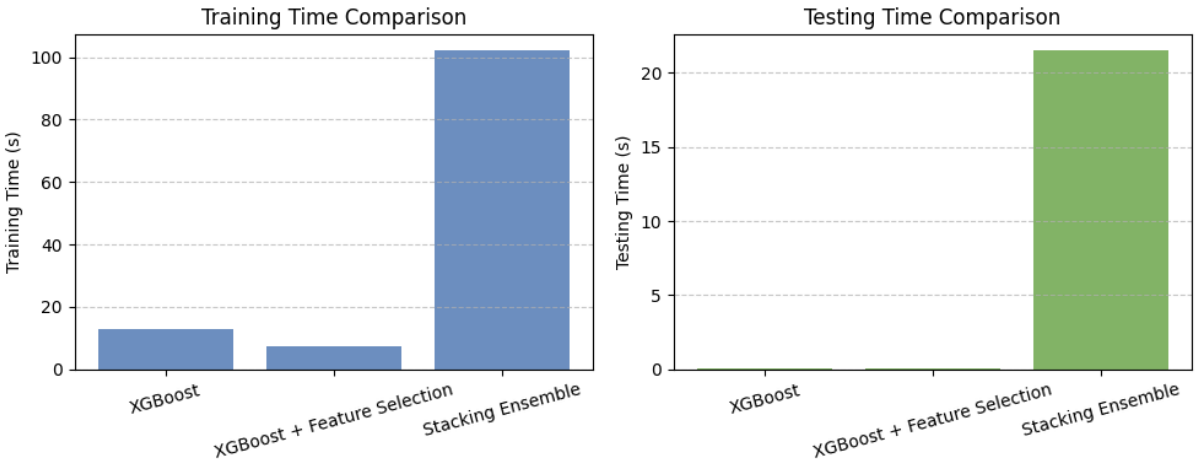

Figure 2. Comparison of training and testing time for XGBoost, XGBoost with feature selection, and the stacking ensemble model

V. Conclusion and Future Works

This study proposed an enhanced Android malware detection framework integrating XGBoost**,** feature selection, and a stacking ensemble. Experimental results on the CICMalDroid 2020 dataset demonstrate that the stacking ensemble outperforms individual classifiers, achieving 98.14% accuracy**,** 93.62% F1-score, and 99.63% ROC-AUC, while feature selection improves computational efficiency by reducing training and testing time. The findings confirm the effectiveness of combining advanced machine learning techniques with comprehensive preprocessing for robust malware detection.

For future work, the framework can be extended by exploring dynamic analysis features in addition to static ones, integrating deep learning architectures such as CNNs or LSTMs, and evaluating the system on larger and more diverse malware datasets. Further research can also focus on real-time deployment of the ensemble model in mobile security applications, optimizing the trade-off between detection accuracy and computational cost.

References

[1] Y. Zhou and X. Jiang, “Dissecting Android malware: Characterization and evolution,” Proc. IEEE Symposium on Security and Privacy**,** San Francisco, CA, USA, pp. 95–109, 2012.

[2] A. Raval and M. Anwar, “Android malware detection: An empirical investigation into machine learning classifiers,” in Proc. 2024 IEEE Int. Conf. on Information Reuse and Integration for Data Science (IRI), 2024, pp. 180–187, doi: 10.1109/IRI62200.2024.00039

[3] F. Saleem, A. R. Javed, F. Iqbal, A. Castiglione, X. Chang, and A. Almomani, “Android malware detection using feature ranking of permissions,” arXiv preprint, arXiv:2201.08468, 2022.

[4] D. B. Ansori, J. Slamet, M. Z. Ghufron, A. Kurniawan, and D. I. Sensuse, “Android malware classification using gain ratio and ensembled machine learning,” Int. J. Safety and Security Engineering, vol. 14, no. 1, pp. 245–254, 2024, doi: 10.18280/ijsse.140126.

[5] C. Palma, A. Ferreira, and M. Figueiredo, “Explainable machine learning for malware detection on Android applications,” Information, vol. 15, no. 1, Art. no. 25, 2024, doi: 10.3390/info15010025.

[6] Wang, X., Zhang, L., Zhao, K., Ding, X., & Yu, M. (2022). MFDroid: A stacking ensemble learning framework for Android malware detection. Sensors, 22(7), 2597.

[7] R. Adriansyah, P. Sukarno, and A. A. Wardana, “Android malware detection using ensemble learning and feature selection with insights from SHAP explainable AI,” in Proc. Int. Seminar on

Intelligent Technology and Its Applications (ISITIA), 2024, pp. 1–6, doi: 10.1109/ISCMI63661.2024.10851666.
[8] E. Alsharif and M. Alharby, "An ensemble machine learning approach for detecting and classifying malware attacks on mobile devices," Arabian Journal for Science and Engineering, 2025, doi: 10.1007/s13369-025-10011-5.
[9] A. Alhogail and R. A. Alharbi, "Effective machine learning-based Android malware detection and categorization," Electronics, vol. 14, no. 8, Art. no. 1486, 2025, doi: 10.3390/electronics14081486.

إطار عمل تراكمي للكشف عن البرمجيات الخبيثة باستخدام XGBoost والغابة العشوائية

محمد م الشح فاطمة أ أبودبوس
قسم علوم الحاسوب، الأكاديمية الليبية للدراسات العليا-مصراتة، ليبيا

استلمت الورقة بتاريخ 2026/01/07، وقبلت بتاريخ 2026/01/23، ونشرت بتاريخ 2026/01/24

الكلمات المفتاحية:
XGBoost، اختيار الميزات، Stacking Ensemble، CICMalDroid 2020، SMOTE، كشف البرمجيات الضارة

الملخص

تتعرض أجهزة Android بشكل متزايد لهجمات البرمجيات الضارة بسبب انتشارها الواسع وطبيعة نظامها المفتوح. تواجه أساليب الكشف التقليدية المعتمدة على التواقيع صعوبة في مواجهة التهديدات الحديثة والمتطورة، مما يستدعي استخدام أساليب ذكية تعتمد على التعلم الآلي. تقدم هذه الدراسة إطارًا محسّنًا لتصنيف البرمجيات الضارة على أجهزةAndroid ، يجمع بين XGBoost و Random Forestلاختيار الميزات، مع استخدام تقنية التجميع .Stacking Ensemble تم معالجة مجموعة بيانات CICMalDroid 2020 من خلال إزالة القيم الشاذة، التحجيم القوي، التحويل الثنائي، وموازنة الفئات باستخدام .SMOTE أظهرت النتائج التجريبية أن إطار التجميع Stacking Ensembleحقق أفضل أداء للكشف، حيث بلغ معدل الدقة 98.14%، و F1-score 93.62%، و99.63% ROC-AUC، متفوقًا على النماذج الفردية. كما ساهم استخدام Random Forestلاختيار الميزات في تحسين الكفاءة الحسابية وتقليل أوقات التدريب والاختبار. تؤكد هذه النتائج أن دمج تقنيات التعلم الآلي المتقدمة يعزز فعالية وكفاءة كشف البرمجيات الضارة على أجهزة.Android