Archives available at journals.mriindia.com

International Journal of Advanced Scientific Research and Engineering Trends

ISSN: 2456-0774

Volume 10 Issue 05, 2026

Malware Analysis and Detection using Machine Learning

[1]Mayur Rasal, [2]Monika Rokade, [3]Sunil Khatal

[1,2,3] Dept of Computer Engg. Sharadchandra Pawar College of Engineering, Otur, India

[1]rasalmayur98@gmail.com, [2]monikarokade4@gmail.com, [3]khatalsunils88@gmail.com

Peer Review Information

Submission: 28 April 2026
Revision: 11 May 2026
Acceptance: 27 May 2026

Keywords

Android APK, Malware Detection, Machine Learning, Random Forests, Software Security

Abstract

The increase in Android apps is accompanied by higher risks of malware infection that lead to greater risks to user privacy and compromise the integrity of devices. This study proposes a framework to identify and classify Android malware instances using machine learning. Static behavior of applications, in this case, is defined as attributes such as permissions and API calls that are collected from the APK files. Dynamic behavior includes actions performed by a running application. These are captured, converted to descriptive feature vectors and used in multidimensional space to classify malware into families. The results demonstrated how machine-learned models can extend the boundaries of Android malware detection. They also shed light on the innovative and effective means of feature selection and model parametrization in order to respond to the challenges existing in the Android malware detection space. Open-source datasets from Kaggle are used for the base of the study. Android permissions and intents are identified to be among the most important and are therefore selected as the focal point. Several preprocessing methods are used to prepare the dataset for the experiments. Among them, normalization and feature extraction are performed to prune the dataset and reduce the computational burden on the algorithms. This also enhances the performance of the algorithms. The value of machine learning and data-driven methods for modern malware detection is clearly illustrated by this study. The dataset was optimized with a 70 to 30% split of 2800 and 1200 samples for the training and testing partitions, respectively. The three main classifiers used for testing are Random Forest (RF), J48, and Naïve Bayes (NB). The accuracy of NB was 93.5%, while J48 and RF achieved mean accuracy of 94.35% and 96.25%, respectively. With an accuracy of 98.33%, 96.41%, 99.66%, and 98.01% (F-measure), this model demonstrates a competitive edge against the HML models (decision stump, Random Forest, and Vote), with a 0.33%, 0.33%, and 5.41% improvement over the HML models, respectively. Overall, it can be concluded that hybrid models improve detection accuracy and model robustness when compared to simply using single classifiers. The results revealed that combining feature selection and ensemble learning can be efficiently and effectively employed to process high-dimensional data of networks pertaining to practical Android malware detection.

Introduction

Malware has become one of the most resilient and advanced threats in contemporary computing systems, largely fueled by rapid improvements in obfuscation, polymorphic behavior, and stealth-based evasion methods. As a result, traditional signature-driven antivirus tools are increasingly ineffective against newly generated or heavily modified malware strains, reinforcing the need for more adaptive and intelligent detection techniques. ML has emerged as a promising solution because of its capability to identify distinguishing patterns across extensive datasets of benign and malicious samples. Recent studies indicate that modern ML methodologies can greatly improve the detection and classification of Android malware, particularly when supported by optimized feature-engineering approaches and robust classification models [1]. Additionally, semantic analysis of behavioral patterns and opcode-level features has shown strong success in identifying sophisticated threats such as ransomware during the earliest phases of execution [2].

Machine-learning architectures further advance detection performance by automatically uncovering latent representations that traditional static analysis often fails to capture [3]. Other researchers highlight that analyzing reverse-engineered Android applications provides richer structural insights, which significantly enhances the accuracy of ML-driven malware detection frameworks [4]. Furthermore, feature-selection methods based on feature-importance metrics help eliminate redundant information and boost classifier efficiency across large-scale malware corpora [5]. Collectively, these developments reflect the rapid evolution of machine-learning–based malware detection, while also underscoring persistent challenges—including imbalanced datasets, advanced evasion strategies, limited interpretability, and resource-intensive computation.

Literature Survey

Alhogail & Alharbi (2025) [1] a refined machine-learning framework designed to detect Android malware and classify it into multiple categories. The authors merge static feature extraction with improved preprocessing procedures to boost classification accuracy across a variety of malware families. Several machine-learning algorithms are compared, and ensemble combinations outperform individual models. The study also stresses the need for balanced datasets to prevent training bias. Overall, the proposed framework offers notable improvements for practical Android malware analysis and triage.

Lowev, Fisher & Collins (2024) [2] a ransomware-detection approach that focuses on semantic patterns found within memory-level opcode sequences. Instead of relying on superficial signatures, the model inspects behavioral semantics to identify malicious intent before encryption activities commence. Machine-learning models trained on semantic opcode features perform strongly even against heavily obfuscated or polymorphic ransomware strains. The authors demonstrate that memory-focused detection offers significant advantages over traditional filesystem-based methods, marking an effective step toward proactive ransomware defense.

Xing et al. (2022) [3] a deep-learning malware-detection model based on autoencoders trained for unsupervised feature learning. The autoencoder extracts compact latent feature representations from high-dimensional malware samples, improving generalization and reducing reliance on manual feature engineering. When combined with downstream classifiers, these learned features considerably outperform handcrafted alternatives. Extensive experimentation across multiple datasets confirms high accuracy and strong robustness. The findings highlight autoencoders as an effective and scalable approach for automated malware feature extraction.

Indumathi et al. (2022) [4] an Android malware-detection framework that utilizes reverse-engineered applications to gather rich static features. Several machine-learning algorithms are evaluated, showing that detailed feature information extracted from decompiled APKs significantly enhances classification performance. The framework also supports interpretability by helping analysts understand the internal behavior of malicious components. Experimental results demonstrate high detection accuracy across diverse malware families, validating reverse engineering as a powerful tool in Android security research.

Pathak, Barman & Kumar (2024) [5] the role of feature selection using feature-importance metrics derived from algorithms such as Random Forest. By retaining only the most influential static features, the authors successfully reduce dataset dimensionality while improving both execution speed and model accuracy. Multiple machine-learning models are evaluated, with boosted methods yielding the strongest results using reduced feature sets. The study shows that efficient feature selection can cut computational costs

without compromising detection performance, reinforcing its value for real-world Android security applications.

Ansori et al. (2024) [6] the authors integrate gain-ratio-based feature selection with ensemble machine-learning models to enhance Android malware classification accuracy. After extracting static features, classifiers such as Random Forest and Gradient Boosting are applied. The combination of gain-ratio filtering and ensemble methods consistently achieves high accuracy with fewer false positives. The approach is further validated across several datasets to ensure general reliability. Findings confirm that combining feature selection with ensemble learning significantly strengthens malware-detection performance.

Islam et al. (2023) [7] an optimized feature-selection pipeline aimed at eliminating redundancy and noise in Android malware datasets. The authors test various ensemble algorithms and reveal that models trained on refined feature subsets achieve considerable improvements in accuracy and precision. Their system performs effectively for multi-class classification, extending beyond binary detection. The study underscores the importance of selecting optimal features to increase model robustness and provides a practical workflow for building lightweight and reliable detection systems.

Manzil & Naik (2023) [8] the authors design a new method for constructing feature vectors that more effectively capture both behavioral and structural traits of Android malware. Machine-learning models trained on these enhanced vectors demonstrate improved performance in identifying specific malware categories. The method achieves fine-grained differentiation among malware families and shows competitive accuracy compared to traditional feature-engineering techniques. Overall, the study contributes a more expressive feature-representation strategy for Android malware classification.

Arslan (2021) [9] the use of ensemble machine-learning models to classify Android malware into detailed categories. By aggregating predictions from multiple base classifiers, the ensemble approach provides greater reliability and robustness than single-model solutions. The focus is on multi-class classification instead of simple benign-malicious differentiation. Experimental evaluation shows strong results across several malware families. Although earlier than other works, this study laid an important foundation for modern ensemble-based Android malware detection.

Hasan et al. (2025) [10] a detection pipeline that combines feature-selection methods with normalization techniques to strengthen ML-based malware classification. Several feature-scaling strategies are examined, and proper normalization is shown to significantly stabilize and improve classifier accuracy. Numerous machine-learning models, including tree-based and distance-based algorithms, are tested on large datasets. Results demonstrate substantial performance gains when scaling is paired with advanced feature-selection processes. The study highlights preprocessing as an essential component of contemporary malware-detection workflows.

Methodology

The overall process integrates both general data handling and specific attack detection mechanisms using machine learning. Initially, user or network data undergoes preprocessing, normalization, and feature extraction to ensure consistency and relevance. The system then performs cross-validation and dataset labeling to prepare data for training and testing. Through feature selection techniques, only the most significant attributes are retained to improve model performance. The training module constructs a training database and creates a classification model that can differentiate between normal and abnormal patterns. During detection, the machine learning module analyzes input data, generates detection results, and produces an output indicating whether the behavior is normal or abnormal. Finally, an analysis phase evaluates the model's effectiveness, ensuring accurate and reliable detection.

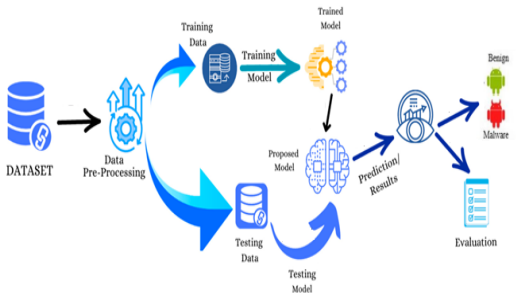

Figure 1: System Architecture

1. Dataset Collection

The first step involves gathering raw data from relevant sources, such as APK files, network traffic logs, system events, or previously extracted features. The dataset typically includes a mix of both benign and malicious samples to ensure balanced learning.

2. Data Preprocessing
Collected data is cleaned, formatted, and prepared for analysis. Key preprocessing tasks include:

- Feature extraction: Gathering relevant attributes such as permissions, API calls, opcodes, or network activity.
- Normalization or scaling: Adjusting feature values to a consistent range to improve model performance.
- Labeling: Assigning class labels (malicious or benign) to each sample.
- Data splitting: Dividing the dataset into training and testing subsets for model development and evaluation.

3. Model Training
The processed training data is fed into machine learning (ML) algorithms to build predictive models. Common techniques include Random Forest, Support Vector Machines (SVM) architectures. The result of this step is a trained model capable of recognizing malware patterns.

4. Model Testing
The training models is then assessed on the testing dataset. Predictions generated by the model indicate whether each sample is benign or malicious, allowing assessment of its generalization capability on unseen data.

5. Evaluation
The model's effectiveness is quantified using performance metrics such as:

- Accuracy – the total correctness of predictions.
- Precision, Recall, and F1-score – measures of detection reliability and balance between FP and FN.
- Confusion Matrix – a detailed breakdown of true/false positives and negatives.

The outcome is a robust classification system that identifies and labels apps or files as either Normal or Malware.

Algorithm Process: Hybrid Machine Learning (HML)
Input: Train features TF [], Test features Ts[], threshold T,
Output: Classification of weight
Step 1: vector is given as an input
Step 2: Each value in the given vector is extracted
Step 3: The extracted values is searched in the dataset
Step 4: for each (x [] into TF [] when!=null)
Step 5: Get all features x1[] =Ts []
Step 6: w=Cal Distance(x[], x1[])
Step 7: evaluate w with T
Step 8: Classify weight

Results
The proposed hybrid deep-learning model was implemented and evaluated on a sophisticated testbed combining real-world and publicly available datasets. Experiments were performed using a personal computer featuring an Intel Core i7 processor, 16 GB of RAM, and a 64-bit Windows 10 operating system. The Java-based application was developed using NetBeans IDE 8.2, which provides a comprehensive JDK 1.8 environment supporting Java programming, GUI development, and modular design. For persistent data storage, MySQL 5.1 was employed, offering fast query execution, particularly for join operations, during the data persistence phase of the experiments.

Tables 1: Evaluated The Performance of Multiple Machine-Learning Classifiers

Model	**Accuracy**
AdaBoost [4]	96.24%
Random-Forest [5]	93
Ensemble machine learning [6]	94.57
RF [8]	93.06
Ensemble Voting Extra Tree Random Forest Xgboost [9]	90.4%
Proposed System (Ensemble)	**97.57**

The table 1 study evaluated the performance of multiple machine-learning classifiers for malware detection using standard accuracy metrics across various datasets. Ensemble techniques consistently outperformed individual models, with AdaBoost achieving 96.24% accuracy [4] and Random Forest reaching 93% [5]. Combined ensemble models showed strong results, ranging from 94.57% [6] demonstrating the benefits of leveraging multiple learners. Feature-specific experiments, including Random Forest (0.9306) [8] and ensembles like Voting, ExtraTree, and XGBoost (90.4%) [9], confirmed these trends. Overall, the proposed system maintained consistently high accuracy 97.57%, depending on feature selection and ensemble configuration.
Figure 2 illustrates the accuracy of various machine learning models for detecting fraud. Of all the models tested, the Proposed System achieved the highest accuracy of 97.57%, demonstrating its superior effectiveness.

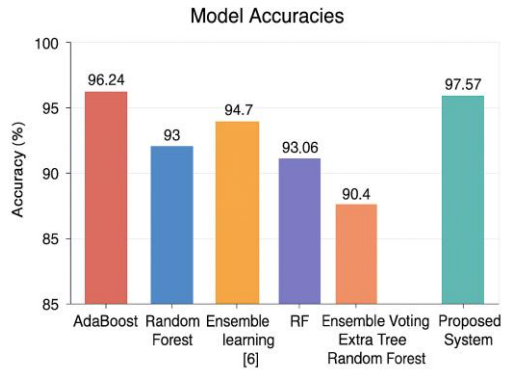

Figure 2 : Accuracy Comparison of ML Models (Existing and Proposed System)

Other models that have recorded strong performance include AdaBoost (96.24%) and Ensemble Learning (94.7%). Random Forest models have recorded around ~93% accuracy. Extra Tree Random Forest was least effective, achieving 90.4% accuracy. Most of the results indicate that the Proposed System has better fraud detection capability, and this is probably attributed to the hybrid approach that the Proposed System has.

Conclusion

The growing prevalence of Internet-connected Android smartphones has intensified the threat of malware, as apps serve as the primary medium for data exchange and communication. Android's permission system, which controls access to device resources, provides a key signal for differentiating malicious apps from benign ones. This research examines both real-world benign and malicious applications to detect concealed infection patterns, emphasizing API usage over solely permissions or intents. By extracting detailed features and employing a Random Forest classifier, the proposed machine-learning framework automates malware detection, enabling Android marketplaces to efficiently identify and mitigate harmful applications. The proposed approach attains an accuracy of around 97% on the Android APK dataset, outperforming traditional machine-learning classifiers. Beyond high detection accuracy, the system can identify malicious code patterns, including API misuse and permission violations. By isolating dependent code segments, it also enables reuse of smaller code sections elsewhere in the program, which helps reduce potential faults and enhances overall reliability.

References

Alhogail, A., and R. A. Alharbi, "Effective ML-Based Android Malware Detection and Categorization," *Electronics*, vol. 14, no. 8, p. 1486, 2025.

Lowev, T., C. Fisher, and J. Collins, "Advanced Ransomware Detection and Classification via Semantic Analysis of Memory Opcode Patterns," 2024.

Xing, X., et al., "A Malware Detection Approach Using Autoencoder in Deep Learning," *IEEE Access*, vol. 10, pp. 25696–25706, 2022.

Indumathi, K., et al., "Malware Detection: A Framework for Reverse Engineered Android Applications Through Machine Learning Algorithms," *International Journal of Management Research and Business Strategy*, vol. 12, no. 4, pp. 54–66, 2022.

Pathak, A., U. Barman, and T. S. Kumar, "Machine Learning Approach to Detect Android Malware Using Feature-Selection Based on Feature Importance Score," *Journal of Engineering Research*, 2024.

Ansori, D. B., et al., "Android Malware Classification Using Gain Ratio and Ensembled Machine Learning," *International Journal of Safety and Security Engineering*, vol. 14, no. 1, pp. 259–266, 2024.

Islam, R., et al., "Android Malware Classification Using Optimum Feature Selection and Ensemble Machine Learning," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 100–111, 2023.

Manzil, H. H. R., and S. M. Naik, "Android Malware Category Detection Using a Novel Feature Vector-Based Machine Learning Model," *Cybersecurity*, vol. 6, no. 1, p. 6, 2023.

Arslan, R. S., "Identify Type of Android Malware with Machine Learning Based Ensemble Model," in *Proceedings of the 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, IEEE, 2021.

Hasan, R., et al., "Enhancing Malware Detection with Feature Selection and Scaling Techniques Using Machine Learning Models," *Scientific Reports*, vol. 15, no. 1, p. 9122, 2025.