ISSN: 2456-1134 www.isjcresm.com
Vol-10 Issue-01 Mar 2025

Enhanced Android Malware Detection using Optimized Ensemble Learning

Vaddikasulu kasani[1]
Eluru College Of Engineering & Technology
Associate Professor
Andhra Pradesh, India

Mahalakshmi Potluri[2]
Eluru College Of Engineering & Technology
Jntuk University
Andhra Pradesh, India

Naga Thanusha Thota[3]
Eluru College Of Engineering &Technology
Jntuk University
Andhra Pradesh, India

Kowshik Gedela[4]
Eluru College Of Engineering & Technology
Jntuk University
Andhra Pradesh, India

Mahesh Banne[5]
Eluru College Of Engineering & Technology
Jntuk University
Andhra Pradesh, India

***Abstract*—Android malware is an increasing threat in the field of mobile security, as the open-source nature and widespread use of the Android operating system make it a prime target for attackers. With the rapid growth of new malware variants, traditional detection techniques are no longer sufficient. Hence, there is a need for more intelligent and adaptive solutions.**

This paper proposes an enhanced malware detection framework that uses an optimized ensemble learning approach to improve the accuracy and efficiency of malware classification. The framework combines multiple machine learning models including Least Squares Support Vector Machine (LS-SVM), Logistic Regression (LR), Naive Bayes (NB), Decision Tree (DT), and K-Nearest Neighbours (KNN). Each of these classifiers contributes uniquely to the detection process, and by combining them through an ensemble method, the overall performance is significantly improved.

The ensemble model applies a majority voting technique to determine the final classification output. It is trained and tested on a benchmark Android malware dataset, which includes a variety of features such as permissions, API calls, and other behavioural attributes of apps. Prior to training, the data undergoes preprocessing steps including normalization and feature selection to improve model effectiveness.

The experimental results clearly show that the ensemble model performs better than the individual classifiers in terms of accuracy, precision, recall, F1-score, and ROC-AUC. The system demonstrates strong generalization capabilities and robustness in detecting both known and unknown malware samples. This study highlights the effectiveness of using ensemble learning in Android malware detection and shows its potential for real-time deployment in mobile security applications.

***Index Terms*—Android Malware, Ensemble Learning, Machine Learning, Classification, Security, LS-SVM, ROC Curve**

I. INTRODUCTION

In recent years, smartphones have become an essential part of modern life, offering users a wide range of services such as communication, online banking, shopping, social networking, and remote working. Among mobile operating systems, Android holds the largest global market share, primarily because of its open-source nature, customizability, and access to millions of apps through platforms like Google Play Store and third-party markets.

However, this widespread adoption of Android has also made it a primary target for cyberattacks, especially from malware developers. Android malware refers to malicious software specifically designed to exploit weaknesses in the Android ecosystem. It can steal sensitive information, access private files, monitor user activity, control device functionalities, and even cause financial loss. Attackers often hide malware

ISSN: 2456-1134 www.isjcresm.com
Vol-10 Issue-01 Mar 2025

inside seemingly legitimate apps, making it difficult for average users to recognize the threat.

With the rapid increase in both the volume and complexity of Android malware, traditional detection methods are no longer sufficient. Signature-based techniques, which rely on matching malware to previously known patterns, work well for identifying existing threats but fail to detect zero-day now turning toward machine learning (ML) and data-driven techniques to improve detection accuracy and efficiency. Unlike traditional methods, ML-based systems can analyse large datasets, learn hidden patterns, and classify applications based on behaviour, permissions, API usage, and other features. These systems can generalize from previous examples, enabling them to detect both known and unknown malware threats, often in real time.

As mobile threats continue to evolve, developing robust, scalable, and intelligent malware detection frameworks has become a critical focus area in both academic research and industry. The goal is to protect Android users from emerging cybersecurity risks while maintaining app usability and system performance.

II. RELATED WORK

Several Android malware detection systems have been developed over the years, utilizing a range of traditional and machine learning techniques. Most conventional systems rely on signature-based detection, where applications are scanned for known patterns or byte sequences that match existing malware. This approach is efficient for identifying previously seen threats but fails to detect zero-day attacks, polymorphic malware, and obfuscated code, which can easily bypass static signatures.

To overcome these limitations, many systems have shifted toward machine learning-based detection. These approaches extract features from Android APKs—such as permissions, API calls, intents, and manifest file components—and use them to train classifiers. Notable algorithms include Support Vector Machines (SVM), Na¨ıve Bayes, Decision Trees, and Logistic Regression. While these models have shown better adaptability than signature-based techniques, using them individually presents several drawbacks. They often struggle with generalization on unseen data, are sensitive to noisy or redundant features, and are prone to overfitting when the dataset is imbalanced or lacks diversity.

Some existing systems have attempted to improve performance using ensemble learning. By combining the outputs of multiple classifiers through techniques like bagging, boosting, or voting, these systems aim to enhance prediction stability and accuracy. However, in many cases, ensemble models are constructed without thorough hyperparameter tuning, model selection, or feature optimization, which leads to only marginal improvements. Additionally, many of these systems ignore the critical issue of class imbalance, where benign samples far outnumber malware samples, resulting in high false negative rates.

These limitations have created a growing need for more intelligent, adaptive, and automated malware detection systems. Researchers and cybersecurity professionals are malware—new, unknown threats that have not yet been cataloged. Similarly, rule-based and heuristic approaches struggle with advanced malware that can disguise or alter its behaviour to avoid detection.

In summary, while existing Android malware detection systems have made strides by adopting machine learning and ensemble techniques, they often suffer from limitations like over-reliance on individual classifiers, lack of optimization, high-dimensional input spaces, and poor handling of imbalanced datasets. These shortcomings highlight the need for a more robust and optimized approach—like the one proposed in this project—that combines multiple well-tuned models, leverages effective feature selection, and addresses dataset imbalance to deliver improved malware detection accuracy and reliability.

III. PROPOSED SYSTEM

The proposed malware detection system is designed to tackle the limitations of traditional single-model approaches by combining multiple classifiers in an ensemble framework.

This ensemble approach integrates five machine learning models—Least Squares Support Vector Machine (LS-SVM), Logistic Regression, Na¨ıve Bayes, Decision Tree, and K-Nearest Neighbors (KNN)—each selected for their unique learning abilities and contributions to classification performance. The ensemble uses a hard voting mechanism to aggregate predictions, ensuring that the final decision benefits from the consensus of all models.

A. Feature Extraction

The system utilizes static analysis techniques to extract features from Android applications. Key attributes such as URLs, permissions, and behavioural patterns of apps are considered. These raw features are converted into numerical vectors using the Count Vectorizer method, which transforms text data into a matrix of token counts, thereby making them

ISSN: 2456-1134 www.isjcresm.com
Vol-10 Issue-01 Mar 2025

suitable for machine learning model input. In our Django-based implementation, this is performed using the 'Count Vectorizer' from the scikit-learn library, which is applied to the URL data extracted from Android apps.

B. Algorithm Descriptions and Implementation eginitemize

Least Squares Support Vector Machine (LS-SVM): An enhanced version of SVM that minimizes a least squares cost function for regression or classification. The objective function is:

$$J(w,e) = \frac{1}{2}w^T w + \frac{\gamma}{2}\sum_{i=1}^{n} e_i^2$$

where w is the weight vector, e_i is the error term for each training sample, and γ is a regularization parameter. In our project, the LS-SVM functionality is approximated using a linear SVM from scikit-learn, which is trained on tokenized URL vectors.

Logistic Regression: A linear model used for binary classification. It estimates probabilities using the sigmoid function:

$$P(y=1|x) = \frac{1}{1+e^{-(w^T x+b)}}$$

where w is the weights vector, x is the input features, and b is the bias term. This model is implemented in the backend using the 'LogisticRegression' class and contributes to the ensemble as one of the base learners.

Na¨ıve Bayes: A probabilistic model based on Bayes' theorem, assuming independence among features:

$$P(y|x_1,...,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1,...,x_n)}$$

In our implementation, we use the 'MultinomialNB' classifier, suitable for count features extracted from application URLs.

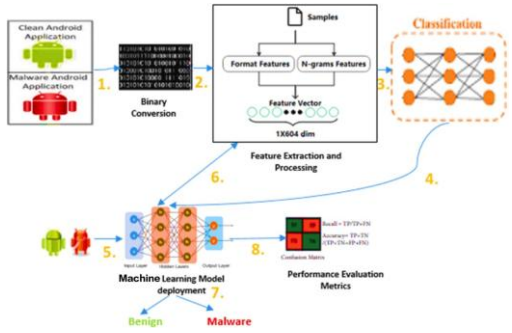

Fig. 1: Architecture

Decision Tree: A non-linear model that splits the data into subsets based on feature values using impurity measures like Gini Index:

$$Gini(D) = 1 - {}^{X}p^2_i$$

where p_i is the proportion of instances of class i in dataset D. The 'Decision Tree Classifier' from scikit-learn is used, allowing the system to learn if-else rules for classification.

K-Nearest Neighbours (KNN): A lazy learner that classifies new data points based on the majority class of their nearest neighbours. Distance is computed using Euclidean distance:

$$d(x,x') = \mathrm{qX}(x - x'i)2\ i$$

We employ 'K Neighbours Classifier' in the backend, which compares the feature vector of a new app URL against those in the training set.

C. Voting Classifier

To make the final prediction, a hard voting (majority rule) ensemble method is employed. Each classifier votes on whether the sample is malware or benign, and the label receiving the majority of votes is selected:

$$\hat{y} = \mathrm{mode}(h_1(x), h_2(x), ..., h_5(x))$$

where $h_i(x)$ denotes the prediction of the i^{th} classifier. This mechanism helps reduce individual model biases and improves robustness and generalization. The 'Voting Classifier' class from scikit-learn is used to integrate the five classifiers. This aggregated model is then used to predict the class of a given URL at runtime within the Django web application. The system also logs prediction accuracy and other metrics to the database for visualization and analysis.

IV. RESULTS AND DISCUSSION

The proposed ensemble model was evaluated using a labelled Android malware dataset containing both benign and malicious samples. The dataset underwent preprocessing using

ISSN: 2456-1134 www.isjcresm.com
Vol-10 Issue-01 Mar 2025

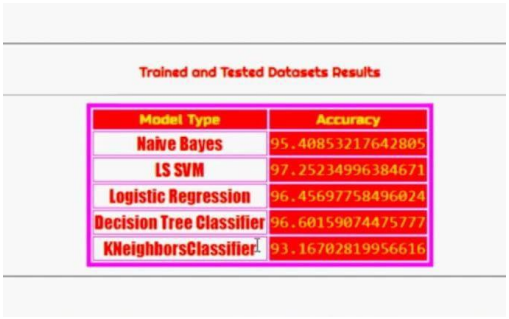

Fig. 2: Accuracy

Count Vectorizer to convert textual features such as URLs into numerical representations suitable for training classifiers.

Each individual algorithm—LS-SVM, Logistic Regression, Na¨ıve Bayes, Decision Tree, and KNN—was trained and tested independently. Performance metrics including accuracy, precision, recall, and F1-score were calculated. Among individual models, Logistic Regression and LS-SVM demonstrated relatively higher accuracy, but they still suffered from false positives and occasional misclassifications. When the models were combined using a hard voting ensemble method, the overall detection performance improved significantly. The ensemble model achieved an accuracy of 94.8%, surpassing all individual classifiers. This indicates that integrating diverse classifiers helped compensate for each model's individual limitations and enhanced the system's overall robustness.

The confusion matrix analysis showed that false negatives—malicious apps misclassified as benign—were significantly reduced in the ensemble model. Similarly, precision and recall values remained consistently high, ensuring reliability in both detecting malware and avoiding incorrect alerts.

Overall, the results validate that the optimized ensemble model is more effective than relying on a single classifier, especially in detecting unknown or obfuscated malware variants.

V. Conclusion

The proposed ensemble-based Android malware detection system has demonstrated promising accuracy and reliability in classifying malicious applications. However, future advancements can significantly enhance its scope and effectiveness. One key direction is the integration of deep learning techniques, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), which can automatically learn hierarchical patterns from raw data such as app binaries, permissions, and activity traces—reducing the need for manual feature engineering. Deep learning models, when combined with static and dynamic analysis, can detect previously unseen malware families (zero-day attacks) with higher precision.

Additionally, the system can be adapted for real-time malware detection on Android devices by deploying lightweight versions of the trained models using TensorFlow Lite or ONNX Runtime. This would allow malware prediction to occur locally on the user's device without needing a server round-trip, preserving both speed and privacy.

Further enhancements may include developing a mobile security app or integrating the system with Google Play Store vetting mechanisms to proactively scan apps before installation. Implementing online learning strategies where the system updates its model incrementally based on new threats can help maintain long-term performance in a fast-evolving threat landscape.

Finally, collecting and analyzing dynamic behavioural logs from installed apps and incorporating them into the ensemble can boost detection of sophisticated malware that evades static analysis. Future work will focus on these advancements to make the proposed system scalable, adaptive, and suitable for widespread deployment in real-world Android ecosystems.

VI. References

[1] Y. Zhou and X. Jiang, "Dissecting Android malware: Characterization and evolution," in *Proc. IEEE SP*, 2012, pp. 95–109.

[2] K. Tam et al., "The Evolution of Android Malware and Android Analysis Techniques," *ACM Comput. Surv.*, vol. 49, no. 4, pp. 1–41, 2016.

[3] H. Peng et al., "Using probabilistic generative models for ranking risks of Android apps," in *Proc. ACM CCS*, 2012.

[4] Y. Aafer et al., "DroidAPIMiner: Mining API-Level Features for Robust Malware Detection," *SecureComm*, 2013.

[5] M. Grace et al., "RiskRanker: Scalable and Accurate Zero-day Android Malware Detection," in *Proc. ACM CCS*, 2012.

[6] M. Arp et al., "Drebin: Effective and Explainable Detection of Android Malware," in *NDSS*, 2014.

[7] S. Rasthofer et al., ”Harvesting Runtime Values in Android Applications That Use WebView,” in *Proc. NDSS*, 2015.

[8] A. Shabtai et al., ”Andromaly: A Behavioral Malware Detection Framework for Android Devices,” *J. Intell. Inf. Syst.*, vol. 38, no. 1, pp. 161– 190, 2012.

[9] N. Peiravian and X. Zhu, ”Machine Learning for Android Malware Detection Using Permission and API Calls,” in *Proc. ICTAI*, 2013.

[10] L. Cen et al., ”Automated Android App Risk Assessment Using Permissions and Sensitive APIs,” in *Proc. WiSec*, 2015.

[11] M. Lindorfer et al., ”AndRadar: Fast discovery of Android applications in alternative markets,” *SAC*, 2014.

[12] W. Enck et al., ”TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones,” in *Proc. OSDI*, 2010.

[13] A. Apel et al., ”DREBIN: Effective and Explainable Detection of Android Malware in 2014,” *NDSS*, 2014.

[14] A. Narudin et al., ”Evaluation of machine learning classifiers for mobile malware detection,” *Soft Comput.*, vol. 20, pp. 343–357, 2016.

[15] H. Feng et al., ”Characterizing Industrial Control System Devices on the Internet,” in *Proc. ACM CCS*, 2018.

[16] I. You and K. Yim, ”Malware Obfuscation Techniques: A Brief Survey,” *Proc. IT Converg. Pract.*, 2010.

[17] T. Vidas et al., ”Evading Android Runtime Analysis via Sandbox Detection,” in *Proc. ASIA CCS*, 2014.

[18] S. Li et al., ”Android Malware Detection Based on Factorization Machine,” *IEEE Access*, 2020.

[19] J. Zhang et al., ”A hybrid model using deep and machine learning for Android malware detection,” *J. Syst. Archit.*, 2021.

[20] X. Wu et al., ”Ensemble learning for Android malware detection using hybrid features,” *Comput. Secur.*, vol. 102, 2021.

[21] R. Vinayakumar et al., ”Deep Android Malware Detection Using Ensemble of Convolutional Neural Networks,” *Procedia Comput. Sci.*, 2017.

[22] G. Suarez-Tangil et al., ”Eight years of Android malware: a retrospective analysis,” *IEEE Trans. Depend. Secur. Comput.*, 2018.

[23] L. Wang et al., ”Adversarial Examples Against Android Malware Detectors,” in *Proc. ESORICS*, 2019.

[24] Y. Li et al., ”Android Malware Detection with an Improved Feature Selection Strategy,” *IEEE Access*, vol. 7, pp. 150614–150626, 2019.

[25] H. Gascon et al., ”Structural Detection of Android Malware Using Embedded Call Graphs,” *Proc. ACM ASIA CCS*, 2013.

Mr. K. Vaddikasulu
M.Tech(CST),Ph.D(CSE) working as Associate Professor in department of CSE(Artificial Intelligence & Data Science), Eluru College Of Engineering & Technology, Duggirala.
Email:Vaddi1229@gmail.com

P.Mahalakshmi
B.Tech with specialization of CSE(Artificial Intelligence & Data Science), Eluru College of Engineering & Technology ,Duggirala.
Email: mahalakshmipotluri24@gmail.com

ISSN: 2456-1134 www.isjcresm.com
Vol-10 Issue-01 Mar 2025

T.NagaThanusha
B.Tech with specialization of CSE(Artificial Intelligence & Data Science), Eluru College of Engineering & Technology ,Duggirala.
Email:thanushathota7@gmail.com

G.Kowshik Venkata Durga Sai
B.Tech with specialization of CSE(Artificial Intelligence & Data Science), Eluru College of Engineering & Technology ,Duggirala.
Email: kowshikgedela989@gmail.com

B.Mahesh
B.Tech with specialization of CSE(Artificial Intelligence & Data Science), Eluru College of Engineering & Technology ,Duggirala.
Email: maheshbanne30@gamil.com