International Journal of Emerging Science and Engineering (IJESE)
ISSN: 2319–6378 (Online), Volume-13 Issue-11, October 2025

Explainable Machine Learning Model for Android Malware Detection

Divish Raj O, Vinay V Hegde

Abstract: ***This study addresses essential cybersecurity challenges in malware detection for applications by developing an explainable machine learning framework. The Stacking Ensemble approach achieves 99.89% accuracy in malware detection while maintaining high explainability through explainable AI (XAI) techniques. The research supports Vector Machines, K-Nearest Neighbours, Logistic Regression, Decision Trees, and Random Forest classifiers with ensemble strategies (Stacking and Voting), used by SHAP and LIME for transparency. The methodology shows that permissions, different API calls, and opcode-related attributes are the features to differentiate malicious applications. Experimental results show that the Stacking Ensemble, which combines individual classifiers across all metrics (accuracy, precision, recall, F1-score), offers a transparent solution for application security that addresses the black-box nature of traditional machine learning models.***

Index Terms: Threat Detection, Explainable AI, SHAP, LIME, Ensemble Learning, Machine Learning, Application Security and Permissions.

Nomenclature:
XAI: Explainable Artificial Intelligence
CNNs: Convolutional Neural Networks
IDS: Intrusion Detection Systems
SMOTE: Synthetic Minority Over-Sampling Technique
LIME: Local Interpretable Model-agnostic Explanations
RBF: Radial Basis Function

I. INTRODUCTION

The use of mobile applications and the increasing complexity of digital ecosystems have heightened the urgency for transparent threat-detection solutions, as Android faces cybersecurity challenges due to its open-source architecture and widespread use. This creates opportunities for attackers but also compels defenders to innovate quickly in the race against advanced malware. Ensemble learning approaches achieve higher classification accuracy than single-model methodologies. Alamro et al. introduced an ensemble-based framework combining machine learning models for Android malware detection, optimising both detection rates and the handling of evolving threats.

Manuscript received on 27 September 2025 | Revised Manuscript received on 06 October 2025 | Manuscript Accepted on 15 October 2025 | Manuscript published on 30 October 2025.
*Correspondence Author(s)
Divish Raj O*, Department of Computer Science and Engineering, R V College of Engineering, Bangalore (Karnataka), India. Email ID: divish727@gmail.com, ORCID ID: 0009-0009-3570-1492
Dr. Vinay V Hegde, Department of Computer Science and Engineering, R V College of Engineering, Bangalore (Karnataka), India. Email ID: vinayvhegde@rvce.edu.in, ORCID ID: 0000-0002-4684-6541

With the success of these detection systems, the adoption of machine learning in cybersecurity introduces new bugs. In exceptional cases, ML-based security models may be susceptible to adversarial attacks that exploit their decision boundaries. Zhang et al. investigated black-box brute-force attack methods that systematically probe and bypass ML-based cyber defences, exposing potential blind spots in widely deployed systems. This research underscores the need not only for stronger models but also for advancing their dependability, trustworthiness, and consistency. These approaches integrate explainable artificial intelligence (XAI) tools, such as SHAP and LIME, to provide understandability and transparency. These XAI methods allow users and analysts to understand and trust the automated decisions made by the detection system. Moreover, it is essential for maintaining confidence and compliance in security-sensitive domains. The rapid iteration of both attack and defence technologies in mobile devices underscores the importance of adaptable, effective security strategies. State-of-the-art research shows how adversarial attacks—ranging from black-box brute-force exploits to code diversification that easily destroys detection and can substantially lower the effectiveness of conventional machine learning models. These tactics expose critical bugs until defence frameworks incorporate not just learning architectures but also explainability and adversarial awareness. It is essential to make a deeper diagnosis of model decisions in security-sensitive environments. As adversarial tactics such as code diversification and black-box evasion become more sophisticated, coupling explainability with resilient architectures becomes increasingly essential for maintaining adequate, accountable security. This cycle of attack and explanation supports breakthroughs in transparent, trustworthy malware defence, ultimately fostering confidence among users, administrators, and the cybersecurity community. These insights shed light on the role of XAI in detection accuracy, especially in evolving threat environments.

II. BACKGROUND AND RELATED WORK

A. Machine Learning for Malware Detection

Machine learning is gaining popularity for detecting malware by analysing its outputs — the structure or behaviour of applications. There are two primary approaches: *static analysis*, which analyses an app's code or configuration files without executing it, and dynamic analysis, which runs the app in a controlled environment to observe its behaviour and runtime. These analyses provide output that models can use to classify applications as benign or malicious.

B. Explainable AI for Malware Detection

SHapley Additive Explanations and Local Interpretable Model-agnostic Explanations are Explainable AI tools that improve transparency in malware detection models by optimising their decision-making processes.

Machine learning algorithms such as Decision Trees, Neural Networks, and K-Nearest Neighbours often yield outcomes that are either malware or not, without evidence. SHAP and LIME help explain these results. SHAP assigns a value to each feature (e.g., permissions or network activity) and quantifies its contribution to the final prediction, providing a global understanding of feature importance. LIME, on the other hand, focuses on individual predictions by giving inputs—such as removing a suspicious API call—and analysing changes in the outcome, to which those inputs are pivotal.

This understandability is essential. For instance, if a Random Forest model identifies an Android app as malicious, SHAP might show that 70% of the decision is due to abnormal data uploads. In contrast, LIME shows that without having such uploads, the app appears benign. This clarity allows security analysts to validate the model's reasoning, detect biases, and improve accuracy. As malware advances evolve—such as the sophisticated banking trojans seen in 2025—combining the power of ML with the transparency of XAI to build trustworthy, advanced malware detection systems.

III. LITERATURE SURVEY

In July 2021, Musikawan et al [4]. were the first to propose AMDI-Droid, a deep learning-based framework designed to advance the detection of Android malware significantly. In their publication, Musikawan et al. described the different functionalities and layers of AMDI-Droid, including its three-layer blending (output) for each hidden layer of the CNN subnetwork and a richer feature-extraction (exploiting) subnetwork. The resulting daemons, like obfuscated malware, were tested against benchmarks. AMDI-Droid outperforms the other benchmarks tested, including static and dynamic analysis. The research work on AMDI-Droid shows that, at a 96% significance level, it achieved high accuracy, precision, and recall. AMDI-Droid is not just precise but also outperforms the benchmarks it was tested against.

This presented (proposed) a novel technique to get a significant enhancement in the detection of Android malware by deploying a framework-based approach rather than training the malware, which is mainly used in traditional malware detection techniques. Also, because DL is used as the malware detection tool, the research paper found that AMDI-Droid outperforms all other detection techniques presented so far.

The complexity of AMDI-Droid and the need to allocate most of the computational power to achieve the required output speed are the only points one should consider before selecting AMDI-Droid as their high-powered detection algorithm.

To replace the traditional malware-detection method, Musicawan et al. proposed AMDI-Droid as the future of Android malware detection in 2011. Proposed the same (model) for the first time in the field, for AMDI-Droid to be non-replaceable, but to serve as a benchmark for a (quite) long-term prediction.

In July 2023, Alamro et al [1]. were the first to present AAMD-OELAC [6] for the detection of Android malware, in contrast to the approach proposed by Musicawan et al., namely AMDI-Droid. Indeed, AAMD-OELAC is still very slow to reach the preset threshold for issuing a judgment on data using the spectrum proposed by Alamro et al. Alamro et al. proposed a different approach to detecting Android malware compared to the proposed method. For me, ensemble learning is the best approach to detecting malware in the near future. In the research work by Alamro et al. on AAMD-OELAC, they deployed an ensemble learning technique that experts in preprocessing noisy datasets have used.

As our favourites are ensemble learning, I would recommend that anyone selecting a malware detection tool choose AMDI-Droid over AAMD-OELAC. As both are based on deep learning, the ensemble learning technique for AMDI-Droid outperforms the method proposed by Alamro et al. Transparency, a cornerstone of this research, is advanced by Explainable AI (XAI) tools such as SHAP and LIME.

[11] In 2022, Alani and Awad introduced PAIRED, a lightweight Android malware detector model built with just 35 static features, including permissions and API calls, and using SHAP for explainability. As PAIRED yields more than 98% accuracy with less resource use, it can be used for real-time detection on any Android device. PAIRED explains the reasons behind flagging an App, such as excessive network access.

In 2022, Liu et al [16]. conducted a significant study to understand the reasons for the high accuracy of the Machine Learning model used for Android malware detection by utilising so-called 'Explainable AI' to uncover the reasons behind it. Surprisingly, they made an interesting discovery. Many of these models achieve inflated accuracy of up to 99%, not because they identify malicious patterns, but because of temporal biases and artefacts in the training datasets.

By conducting a comprehensive XAI analysis across seven datasets, they have reported findings that reveal that, in most cases, the models tend to learn different patterns in the dataset rather than detecting generalisable characteristics of malware. For instance, some of the essential features learned by the model could, in fact, be different temporal indicators that are guaranteed to occur when the samples are collected but cannot be proven to be actually malicious. The paper serves to attract the attention of other researchers, always to cross-check their model performance with/without explainable AI techniques, as well as the datasets used.

The artefacts. Proposing their findings emphasises the need for rigorous evaluation methodologies and transparent reports on how and why models make predictions. This ensures that detection systems would remain effective when deployed in production environments.

Kirubavathi and Nithish [12] adopted a dynamic approach in 2024. They combined Random Forest, XGBoost, LightGBM, and KNN with a Random Forest meta-learner. The CCCS-CICAndMal2020 dataset was used for implementation. LIME improved its framework, achieving 98% accuracy. Class balancing was performed using SMOTE.

Feature ranking was done to reduce noise. They have proposed a transparent, high-performance malware detection system.

Smmarwar et al. [13] presented XAI-AMD-DL in 2023. This is a deep learning approach to designing an explainable AI for an Android malware detection system. This model combines a Convolutional Neural Network (CNN) with Bi-Gated Recurrent Units (Bi-GRU). This hybrid network can simultaneously learn the malware's spatial and temporal behaviours. Many deep learning models fail to provide explainability. So the conventional deep learning models cannot solve the problem. They have designed their deep learning system with built-in methods to make it explainable. This architecture was tested on standard benchmark datasets. Their hybrid model outperformed both traditional Convolutional Neural Networks (CNNs) and RNNs. In addition, since they have conducted feature contribution analysis, it provides some transparency to security professionals, classifying their approach as the most reliable Android malware detection model.

Rajalakshmi et al. [14] paved the way in 2024. They made memory analysis their core approach, combined with explainable AI, to improve their malware detection system. Their conventional malware detection method showed a high false-positive rate and was therefore taken in an innovative direction; this work emphasises that traditional malware detection methods show high accuracy. However, it fails to detect versions of obfuscated malware because it cannot access the file's memory.

Two authors of the present study, Patel and Ghosh [15], presented AMD-XAI-ML, a machine learning-based Android malware detection framework designed with explainability-oriented smart computing aspects in 2024. Evasion techniques like packing and encryption pose significant challenges for creating an effective malware detection system. The AMD-XAI-ML framework is well-engineered to detect unknown malware variants while maintaining low false-positive rates. Nine machine learning models are used to compute AMD-XAI-ML on the CICAndMal2019 dataset. Among all, the XGBoost function provides the highest accuracy of 98.54%, while the random forest classifier offers 98.42%. It is seen that the decision tree classifier also computes the best accuracy of about 98.23%.

Explainability can be achieved using XAI (eXplainable Artificial Intelligence) techniques. So, the transparency of the decision-making is well balanced with the performance of the detection process. The cause plays a significant role in smart devices that require minimal resources. The machine learning model's explanation makes it easier for security analysts to determine whether malware is present. For practical deployment, a transparent model for detecting Android malware is feasible and accurate.

Moustafa et al. gave a brief overview of the role of XAI in IoT intrusion detection in 2023 [7]. In the different models used to implement machine learning methods for IoT intrusion detection, an interpretable model is needed to explain the predictions. Overfitting is often a significant issue in intrusion detection systems (IDS). So, trustworthiness is also one of the key outcomes to be achieved by improving machine learning methods and techniques.

An efficient defence strategy is also needed to support plans to develop a better IoT model. The real question about the bug is whether there was interest in using SHAP and LIME as an operational technique for IoT botnets in May 2024. The actual questions are the botnet prediction model's loyalty and sensitivity to the training data. So, one can conclude that the bug in the Android malware is correctly explained. The SHAP is providing the highest loyal reasoning for the prediction. The SHAP is also accurate for the decision reached, as the botnet traffic pattern matched that of the Android malware. The model provides satisfying explanations and precise predictions.

Kalakoti et al. proposed an operational technique for applying SHAP and LIME to the IoT botnet in 2024. The bug in the Android malware yields high predicted and correct results when predicting the botnet traffic pattern. The real decision for the bug in the malware is matched, and SHAP is used for faithful or loyal reasoning to predict the traffic pattern [8]. Kalakoti et al. contributed to the world of IOT BOTNET in May 2024.

Chandana Snehal et al [17]. documented the proven, unofficial results that it matches the proposed machine learning and deep learning models for Android malware in 2024. The unexplored intrusion project is revealed by Chandana Snehal et al. The botnet traffic pattern matched the Android malware pattern, and SHAP is accurate. Kalakoti et al. contributed to the use of botnet traffic patterns and their performance in the operation of the Android malware. Kalakoti and named their system to detect IoT botnets as XAI. The entire process is designed to combine machine learning (Random Forest) and deep learning (LSTM-CNN, BiLSTM) with SHAP for detecting Android malware. Liverpool J also used the LSTM-CNN with SHAP in October 2023 [3] to operate the botnet and detect Android malware. This proposed hybrid model combines the SMOTE technique for class-imbalanced datasets and achieves accurate classification of Android malware by leveraging better suspicious API usage [8]. Officials made botnet traffic detection easier, making the entire process easier to detect and act on. While scalability is the primary advantage, there are also specific concerns. Atedjio et al. [5] in 2024 developed a CycleGAN-based defence method that included a gradient penalty to improve stability and accuracy. It was introduced for the grey-box setting where the attacker has only partial knowledge about the detection model. In April 2024, Xu et al. [10] reported, in one of their research papers on OFEI, a semi-black-box attack in which they modified one feature per iteration, achieving a 98.25% evasion rate. Their Bayesian-uncertainty-based countermeasure detected 99.28% of the above iterative evasion attacks, highlighting the increasingly vulnerable future landscape. Shu and Yan [6] created EAGLE in July 2024, a method that uses LIME-based explanations to prevent Android classifiers from achieving 92.4%-100% success (33). The transferability of the attack across different models is alarming, highlighting how the concept of eXplainable AI (XAI) can enhance transparency while also widening the attack surface. Zhang et al. [2] in 2020 proposed another brute-force attack, the BFAM black-box attack, was proposed, which uses the

confidence scores and the attack rate of the proposed detection model.

IV. IV. METHODOLOGY

A. Dataset Description and Preprocessing

The TUANDROMD dataset serves as the primary resource for the study, giving a publicly available collection of Android application data. It has a diverse set of benign and malicious applications determined through both static and dynamic analysis. The dataset includes features, such as *permissions*, which have the access requests made by an application; *API calls*, which give the system functions called by the application; *opcodes*, referring to the low-level instructions executed; and *network features*, which provide the application with its communication behaviour and data transmissions over the internet and security

B. Data Cleaning and Preprocessing

To prepare the dataset for machine learning, multiple preprocessing steps are applied, including imputing numerical values with the median and filling categorical values with the mode. Numerical features are normalised using Min-Max Scaling to handle different input ranges. Due to the class imbalance—with more benign than malicious applications—Synthetic Minority Over-sampling Technique (SMOTE) is used to generate synthetic malware samples and balance the dataset. The dataset is split into training and test sets at 80:20, with stratified sampling to maintain the same class distribution in each subset.

C. Feature Selection for Optimised Model Performance

To improve model performance and classification accuracy, feature selection is performed before training. Two approaches are used to identify the suitable features for malware detection.

The first approach involved training a Random Forest model and using Shapley Additive Explanations values to evaluate feature importance. The top 50 features are selected based on their mean absolute SHAP values. In the 2nd approach, feature importance scores are directly taken from a Random Forest classifier, and the 50 highest-ranked features are considered for model training.

Both approaches show that permissions, API calls, and opcode-related attributes are the most important for distinguishing between benign and malicious applications and for reducing the dataset to features that improve performance and yield higher model performance and classification accuracy.

D. Machine Learning Models and Ensemble Learning for Malware Detection

This study used different machine learning algorithms and ensemble techniques to improve malware detection performance. The models are classifiers such as Support Vector Machines, K-Nearest Neighbours, Logistic Regression, Decision Trees, and Random Forest.
Forests.

SVMs are trained using various kernel functions—linear, polynomial, and radial basis function (RBF)—to capture both complex and straightforward decision boundaries. KNN classification works by examining the k nearest neighbours in the feature space. Logistic Regression modelled the probability that an application is malicious. Decision Trees generated rule-based decisions through recursive partitioning, and Random Forests used multiple decision trees to improve prediction and stability.

To further improve predictive accuracy, ensemble learning techniques such as *Stacking* and *Voting* are implemented. In Stacking, individual base learner models (SVM, KNN, Decision Tree, and Random Forest) are trained separately, and a Logistic Regression model acts as the meta-learner to combine their predictions. Voting ensembles aggregated predictions from all classifiers—*hard voting* selected the majority class, while *soft voting* averaged class probabilities to get the final output.

These models were designed to strengthen individual learners and improve the quality, dependability, trustworthiness, and consistency of malware detection systems.

E. Explainability and Feature Importance Analysis Using SHAP and LIME

To improve the understandability of model predictions and address the black-box nature of complex classifiers, Shapley Additive Explanations and Local Interpretable Model-agnostic Explanations are used.

i. *Global Feature Importance Using SHAP:* SHAP is used to categorise the overall contribution of each feature to the model's decision-making process. By calculating SHAP values across all test samples, global feature importance is calculated. The importance score for a given feature f_i is calculated as the mean of the absolute SHAP values over all N test instances, as proved below:

$$\text{Importance}(f_i) = \frac{1}{N}\sum_{j=1}^{N} |\text{SHAP}_{j,i}| \quad (1)$$

Here, $\text{SHAP}_{j(i)}$ represents the SHAP value of feature f_i for the j^{th} sample.

ii. *Local Instance-Based Understandability Using LIME:* LIME is used to explain individual predictions. It generates a dataset X' by altering the input features and fits a surrogate model g, typically a linear model, to approximate the behaviours of the original model f: $\hat{y} = g(X', \lambda)$ (2)

where $\hat{y}$ is the predicted outcome, λ is a locality-aware weighting function, and X' contains the instances. This local explanation helps understand how specific features influenced the model's decision for a given sample.

F. Feature Correlation Analysis

A correlation heatmap is generated to analyse feature interdependencies. When two features show high correlation, one could be removed to reduce redundancy. The Pearson correlation coefficient between features X_i and X_j is defined as:

$$\rho(X_i, X_j) = \frac{\sum_{k=1}^{N}(X_{i,k} - \bar{X}_i)(X_{j,k} - \bar{X}_j)}{\sqrt{\sum_{k=1}^{N}(X_{i,k} - \bar{X}_i)^2}\sqrt{\sum_{k=1}^{N}(X_{j,k} - \bar{X}_j)^2}} \quad (3)$$

Where $\bar{X}_i$ and $\bar{X}_j$ are the mean values of features X_i and X_j, respectively, this analysis ensured that only informative and non-redundant features are red, boosting model performance.

G. Performance Evaluation

We computed how effectively various machine learning models identified Android malware by determining key metrics such as accuracy, precision, recall, and F1-score. Among the traditional models—Support Vector Machines with Linear, RBF, and Polynomial kernels, K-Nearest Neighbours, Decision Tree, and Logistic Regression—each contributed approximately 11.0% to 11.1% in overall performance.

The Stacking Ensemble method, in particular, is good, as reflected in the comparative performance pie chart, achieving the highest combined scores across all metrics. This shows its ability to merge the strengths of multiple base classifiers and enhance a meta-classifier, thereby improving predictive accuracy and efficiency. It confirms that different base learner models and meta learner models, and their outcomes through a final learning step, provide a better malware detection strategy.

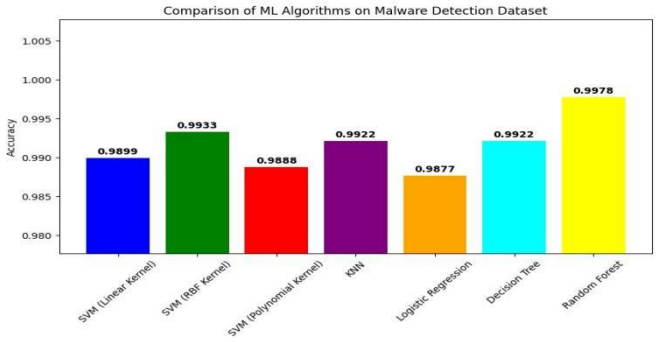
The ensemble learning methods demonstrated superior consistency and interpretability for essential applications, such as Android malware detection.

V. RESULTS AND DISCUSSION

This section presents the experimental results and discusses the determination of the proposed malware detection framework, including performance evaluation, feature importance analysis (SHAP and LIME), and explainability. Existing research has demonstrated the effectiveness of machine learning for detecting Android malware. The accuracy of models depends on the choice of features, the quality of data, and the complexity of the machine learning model. A multi-approach combining both static and dynamic features has proved more effective than relying on either type alone.

A. Model Performance Visualisation

Figure 1 presents a bar chart comparing the accuracy of various machine learning models in detecting Android malware. The models are Support Vector Machines with Linear, RBF, and Polynomial kernels; K-Nearest Neighbours; Logistic Regression; Decision Tree; and Random Forest. Among these, Random Forest achieved the highest accuracy of 0.9978, followed closely by Decision Tree at 0.9922. The SVM with an RBF kernel also demonstrated strong performance, recording an accuracy of 0.9933. In contrast, Logistic Regression showed the lowest accuracy at 0.9877. This shows how ensemble-based models, such as Random Forest and Decision Tree, outperform individual classifiers in malware detection tasks. Handling complex patterns and noise is significant, making them more preferable for essential applications.



[Fig.1: Comparison of Model Accuracy for Android Malware Detection Across Different Classifiers]

Table I summarizes the classification performance of different machine learning models applied to the TUANDROMD dataset. The performance of each model is computed using four metrics: accuracy, precision, recall, and F1-score. The results show that ensemble models (Stacking and Voting) outperformed individual classifiers, achieving the highest accuracy and quality.

Table I: Classification Performance of Machine Learning Models on Tuandromd dataset

Model	Accuracy	Precision	Recall	F1-Score
SVM (Linear)	0.9899	0.9873	0.9812	0.9842
SVM (RBF)	0.9933	0.9876	0.9916	0.9896
SVM (Polynomial)	0.9888	0.9754	0.9909	0.9829
KNN	0.9922	0.9909	0.9847	0.9877
Logistic Regression	0.9877	0.9838	0.9778	0.9807
Decision Tree	0.9922	0.9951	0.9806	0.9876
Random Forest	0.9978	0.9986	0.9944	0.9965
Stacking Ensemble	0.9989	0.9993	0.9972	0.9983
Voting Ensemble	0.9966	0.9958	0.9937	0.9948

B. Performance Distribution Across Models

The performance distribution among the computed models—SVM (Linear, RBF, Polynomial), KNN, Logistic Regression, Decision Tree, Random Forest, Stacking Ensemble, and Voting Ensemble—is remarkably balanced, with each contributing roughly 11.0%–11.2% to the overall accuracy, precision, recall, and F1-score metrics. This even spread shows that no single model overwhelmingly dominates across all criteria. However, the Stacking Ensemble slightly outpaces the rest at 11.2%, indicating its effectiveness at aggregating diverse classifiers to improve generalisation.

It is observed that each model occupies a roughly equal slice, ranging from 11.0% to 11.2%, indicating a balanced evaluation across performance metrics. The Stacking Ensemble stands out slightly at 11.2%, indicating it edges out the others when all metrics are considered together. Random Forest and Decision Tree also hold strong positions at 11.2% and 11.1%, respectively, while SVM (Polynomial) and KNN sit at the lower end with 11.0%.

This performance spread indicates model optimisation, but it also signals diminishing returns from incremental improvements within individual models. The Stacking Ensemble model achieved the highest accuracy, followed by the Voting Ensemble, demonstrating the advantage

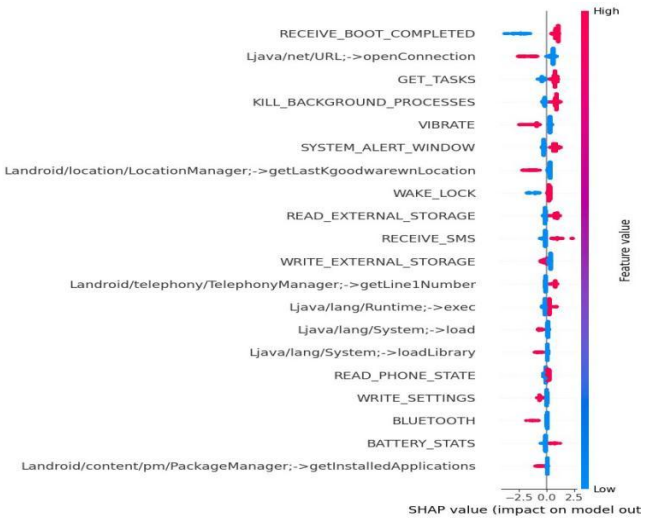
of combining multiple classifiers. Random Forest and Decision Tree also perform robustly, enabling them to handle complex feature interactions and non-linear patterns. Meanwhile, models like SVM (Polynomial) and KNN, though marginally behind, still maintain strong performance, reflecting the total competitiveness of traditional models in malware detection contexts.

i. *Analysis of Performance:* Random Forest achieved the highest accuracy at 99.88%, showing its knack for handling malware classification. The Bagging Classifier is not far behind, reaching 99.77%, further demonstrating the effectiveness of ensemble learning methods. XGBoost and CatBoost achieved slightly lower accuracies of 99.32% and 99.44%, respectively, but still showed strong performance in detecting malware.

These results show that ensemble methods—particularly those based on bagging, such as Random Forest and Bagging Classifier—outperform boosting-based models in this specific dataset. The findings show the quality and strength of bagging ensembles when applied to the malware detection task.

C. Feature Importance Analysis

Figure 2 visualizes the feature importance taken from a Random Forest classifier. The bar chart arranges features by their calculated importance values, with the most influential features along the x-axis and their corresponding feature names on the y-axis. This visualisation helps identify which features have the most significant impact on the model's decision-making.



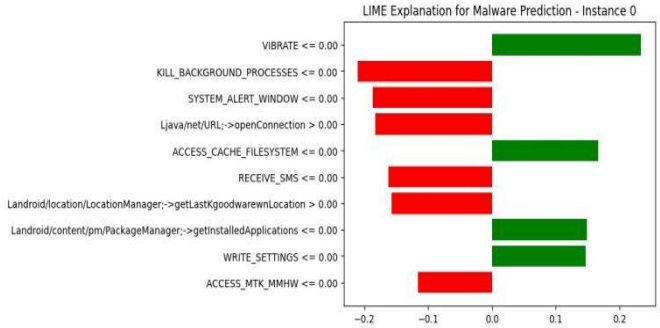
[Fig.2: SHAP-Based Feature Selection]

D. LIME Prediction Analysis

Figure 3 presents the Local Interpretable Model-agnostic Explanations (LIME) analysis for a specific benign application instance. The model demonstrated 100% confidence in its benign classification. Key contributing features include the absence of permissions such as VIBRATE, KILL BACKGROUND PROCESSES, and SYSTEM ALERT WINDOW, all of which had values of 0, determining that the app does not access these potentially intrusive functionalities.

Additional permissions, such as RECEIVE SMS and ACCESS CACHE FILESYSTEM, are also disabled, further supporting the benign classification. While the app makes specific API calls such as *Ljava/net/URL; →openConnection* and *Landroid/location/LocationManager;*

LIME's explainability provides a clear rationale for the model's decision-making process, confirming that the absence of typical malicious indicators yields a high-confidence benign prediction.



[Fig.3: LIME Explanation for Malware Prediction – Instance 0]

E. SHAP Explainability Analysis

To ensure our model isn't just a mysterious black box, we used Shapley Additive Explanations to understand what's driving its decisions and how each feature contributes. SHAP gives us the complete clarity of what's pushing the model to flag an app as malware or not.

Figure 4 shows a SHAP waterfall plot that breaks down which features tipped the scales for a specific malware call. The heavy features pushing toward a malware label are *RECEIVE SMS*, *READ EXTERNAL STORAGE*, and *RECEIVE BOOT COMPLETED*—these are the kind of permissions malware loves to grab for sneaky access and sticking around after a reboot. On the flip side, stuff like *SYSTEM ALERT WINDOW* and *KILL BACKGROUND PROCESSES,* the other way, but their pull is lower.

SHAP enables a detailed understanding of how different app attributes influence the final prediction—whether the model flags an app as malicious or safe. As a traditional feature importance metric, SHAP assigns a value to each feature based on cooperative game theory, ranking them by how much they push the prediction toward or away from "malware."

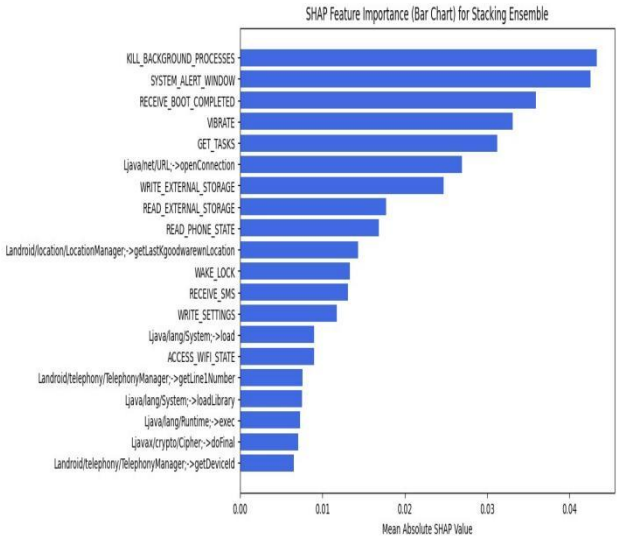
This kind of insight is crucial. It not only improves trust in high-performing models such as Stacking Ensemble and Random Forest, but also supports developers and security analysts in verifying whether the flagged behaviours actually make sense, ensuring that alerts are both actionable and interpretable. In most evolving threats, such as Android malware in 2025, this transparency is a technical and operational requirement.

Retrieval Number:100.1/ijese.L261713111025
DOI:10.35940/ijese.L2617.13111025
Journal Website: www.ijese.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

[Fig.4: SHAP Waterfall Plot for Individual Malware Classification]

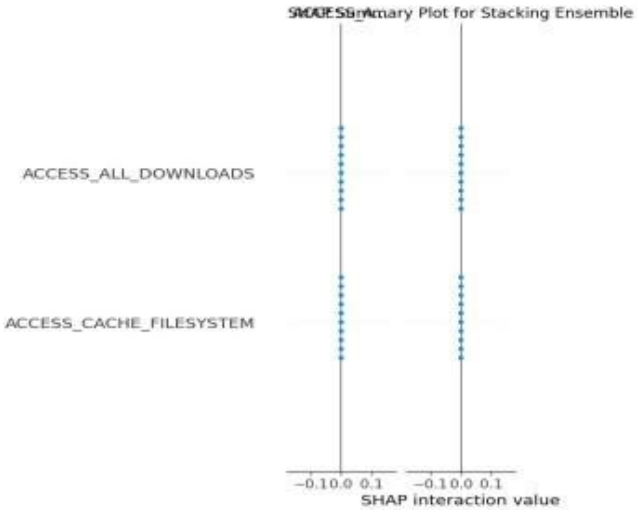
Figure 5 presents the SHAP feature importance bar chart for the stacking ensemble model, highlighting the most influential features contributing to malware classification. At the top of the list is *KILL BACKGROUND PROCESSES*, which is commonly associated with malicious behaviour aimed at interfering with background processes to avoid detection. Other essential features are *SYSTEM ALERT WINDOW* and *GET TASKS*, both of which are frequently exploited by malware to manipulate system-level functionalities.



[Fig.5: SHAP Feature Importance Bar Chart for Stacking Ensemble Model]

Figure 6 presents the SHAP summary plot, illustrating how various features interact and how their SHAP values vary across different instances. Features such as *ACCESS ALL – DOWNLOADS* and *ACCESS CACHE FILESYSTEM* exhibit significant variability, indicating their context-dependent influence on the model's predictions. These findings support the notion that permission-based features are central to effective Android malware detection, aligning with previous research on malicious application behaviour.

Collectively, the SHAP visualisations underscore the critical role of permissions in identifying threats and validate the emphasis placed by the stacking ensemble model. What distinguishes SHAP in this context is not only its capability to rank features by importance but also its ability to reveal when and how a feature contributes. This facilitates a deeper, more interpretable understanding of the model's decision-making process, bridging the gap between complex machine learning outputs and human-explainable insights.



[Fig.6: SHAP Summary Plot Showing Feature Interactions]

The LIME analysis confirms that specific Android permissions significantly impact malware classification. The ability to interpret model decisions through explainable AI techniques improves trust and dependability and enhances the consistency of automated malware detection systems. This analysis integrates SHAP-based global feature explanations with LIME-based local insights to create a more robust framework for understandability. The results suggest that while LIME gives local feature contributions for an individual prediction, SHAP offers a more comprehensive analysis by evaluating feature importance at both regional and global levels.

F. SHAP Heatmap Analysis

i. SHAP Correlation Heatmap: The correlation heatmap shown in Figure 7 visualises the pairwise relationships between SHAP values of different features, offering insights into how these features influence the model's predictions. Positive correlations (highlighted in red) indicate features that contribute similarly to malware classification, while negative correlations (highlighted in blue) reveal inverse relationships in their SHAP impacts.

For instance, features such as *ACCESS LOCATION EXTRA COMMANDS* and *AC-*

CESS SHARED DATA display a strong positive correlation, indicating they often co-occur in malware samples and jointly influence the prediction outcome. Conversely, weak or negative correlations may point to independent or opposing roles in the model's decision logic.

This correlation map is particularly valuable for identifying feature dependencies and potential redundancies, enabling further optimisation of the feature set used in the malware detection model.

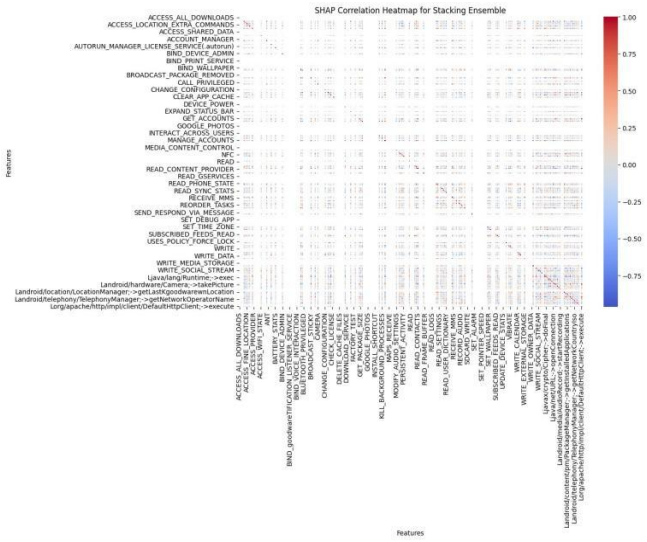
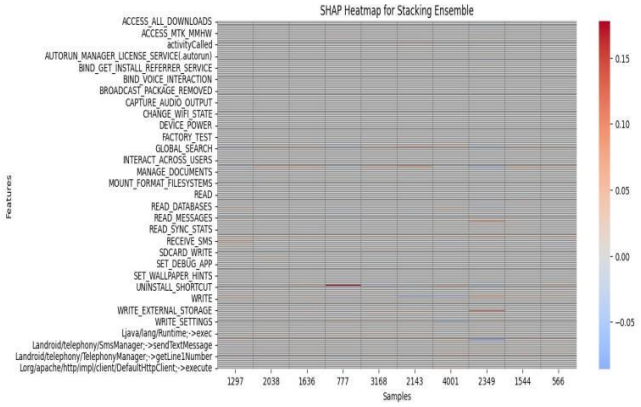


Fig.7: SHAP Correlation Heatmap Showing Feature Relationships]

ii. *SHAP Feature Importance Heatmap:* The heatmap presented in Figure 8 illustrates SHAP values across numerous individual samples, enabling a granular and straightforward understanding of feature contributions in the malware classification task. Features with similarly high SHAP values across samples show strong influence on prediction outcomes, whereas features with near-zero or neutral values have less impact.

This outcome aids explainability by clearly identifying which permissions or behaviours are most responsible for the classification decisions made by the stacking ensemble model. For example, permissions related to writing to external storage or interacting with high-level system APIs result in higher SHAP values, indicating their substantial influence in flagging malicious behaviour.



[Fig.8: SHAP Feature Importance Heatmap for Stacking Ensemble. This Heatmap Illustrates SHAP Values Across multiple Samples, Highlighting the Most and Least Contributing Features in Malware Detection]

VI. CONCLUSION

This study proposed an explainable ensemble-based approach for Android malware detection, integrating SHAP and LIME to improve model understandability. Using SHAP-based feature selection, the most influential permissions and API calls are identified, thereby improving model performance. Among the computed models, the ensemble techniques—particularly the Stacking and Voting classifiers—have shown superior accuracy and quality compared to individual classifiers. The inclusion of explainability tools yielded meaningful insights into model behaviour, thereby increasing transparency and enabling automated cybersecurity systems.

FUTURE WORK

Future work will explore expanding the range and volume of datasets to improve the process across a broader spectrum of malware and variants. More advanced explainability techniques, such as DeepSHAP and multi-explainable AI frameworks, will be introduced in the future to improve the understandability of complex deep learning models. Real-time deployment strategies for mobile environments will also be developed, with a focus on performance, accuracy and cyber threats.

DECLARATION STATEMENT

After aggregating input from all authors, I must verify the accuracy of the following information as the article's author.

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Funding Support:** This article has not been funded by any organizations or agencies. This independence ensures that the research is conducted with objectivity and without any external influence.
- **Ethical Approval and Consent to Participate:** The content of this article does not necessitate ethical approval or consent to participate with supporting documentation.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Author's Contributions:** The authorship of this article is contributed equally to all participating individuals.

REFERENCES

1. H. Alamro, W. Mtouaa, S. Aljameel, A. S. Salama, M. A. Hamza, and A. Y. Othman, "Automated Android Malware Detection Using Optimal Ensemble Learning Approach for Cybersecurity," *IEEE Access*, vol. 11, pp. 72509-72517, 2023. DOI: http://doi.org/10.1109/ACCESS.2023.3294263
2. S. Zhang, X. Xie, and Y. Xu, "A Brute-Force Black-Box Method to Attack Machine Learning-Based Systems in Cybersecurity," *IEEE Access*, vol. 8, pp. 128250-128263, 2020. DOI: http://doi.org/10.1109/ACCESS.2020.3008495
3. M. B. Mwangi and S. M. Cheng, "An Adversarial Attack on ML-Based IoT Malware Detection Using Binary Diversification Techniques," *IEEE Access*, vol. 12, pp. 175563-175578, 2024. DOI: http://doi.org/10.1109/ACCESS.2024.3506841
4. P. Musikawan, Y. Kongsorot, I. You, and C. So-In, "An Improved Deep Learning Neural Network for the Detection and Identification of Android Malware," *IEEE Access*, vol. 11, pp. 115475-115487, 2023. DOI: http://doi.org/10.1109/ACCESS.2023.3324524
5. F. S. Atedjio, J.-P. Lienou, F. F. Nelson, S. S. Shetty, and C. A. Kamhoua, "CycleGAN-Gradient Penalty for Enhancing Android Adversarial Malware Detection in Grey Box Setting," *IEEE Access*, vol. 12, pp. 42513-42526, 2024. DOI: http://doi.org/10.1109/ACCESS.2024.3377938
6. Z. Shu and G. Yan, "EAGLE: Evasion Attacks Guided by Local Explanations Against Android Malware Classification," *IEEE Transactions On Dependable and Secure Computing*, vol. 21, no. 6, pp. 5436-5451, Nov.-Dec. 2024.

DOI: http://doi.org/10.1109/TDSC.2024.3363446
7. N. Moustafa, N. Koroniotis, M. Keshk, A. Y. Zomaya, and Z. Tari, "Explainable Intrusion Detection for Cyber Defences in the Internet of Things: Opportunities and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1775-1807, third quarter 2023. DOI: http://doi.org/10.1109/COMST.2023.3280465
8. R. Kalakoti, H. Bahsi, and S. Nomm, "Improving IoT Security With Explainable AI: Quantitative Evaluation of Explainability for IoT Botnet Detection," *IEEE Access*, vol. 12, pp. 85648-85666, 2024. DOI: http://doi.org/10.1109/ACCESS.2024.3413615
9. A. Rashid and J. Such, "MalProtect: Stateful Defence Against Adversarial Query Attacks in ML-Based Malware Detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 3165-3179, July-Aug. 2023.
DOI: http://doi.org/10.1109/TDSC.2022.3194598
10. G. Xu, G. Xin, L. Jiao, J. Liu, S. Liu, M. Feng, and X. Zheng, "OFEI: A Semi-Black-Box Android Adversarial Sample Attack Framework Against DLaaS," *IEEE Access*, vol. 12, pp. 59673-59688, 2024.
DOI: http://doi.org/10.1109/ACCESS.2024.3392823
11. M. M. Alani and A. I. Awad, "PAIRED: An Explainable Lightweight Android Malware Detection System," *IEEE Access*, vol. 10, pp. 73214-73228, 2022.
DOI: http://doi.org/10.1109/ACCESS.2022.3189390
12. G. Kirubavathi and S. Nithish, "Dynamic Ensemble Learning Framework Improved with XAI To Detect Android Malware," in *Proc. 2024 International Conference on Intelligent Systems for Cybersecurity (ISCS)*, Dehradun, India, May 2024, pp. 1-6.
DOI: http://doi.org/10.1109/ISCS61804.2024.10581285
13. S. K. Smmarwar, G. P. Gupta, and S. Kumar, "An Explainable AI Approach for Android Malware Detection Using Deep Learning Techniques," in *Proc. 2023 IEEE World Conference on Applied Intelligence and Computing (AIC)*, Sonbhadra, India, July 2023, pp. 467-472. DOI: http://doi.org/10.1109/AIC57670.2023.10263905
14. R. Rajalakshmi, S. Pallavi, U. Naresh, S. Telang, and S. Kiran, "Advancing Malware Detection Using Memory Analysis and Explainable AI Approach," in *Proc. 2nd International Conference on Intelligent Cyber Physical Systems and IoT (ICoICI)*, Chennai, India, Nov. 2024, pp. 463-468.
DOI: http://doi.org/10.1109/ICoICI58642.2024.10486624
15. A. Patel and S. M. Ghosh, "Android Malware Detection Using Explainable Machine Learning for Secure Computing," in *Proc. 2024 OPJU International Technology Conference (OTCON)*, Raigarh, India, June 2024, pp. 1-6.
DOI: http://doi.org/10.1109/OTCON60325.2024.10687498
16. Y. Liu, C. Tantithamthavorn, L. Li, and Y. Liu, "Explainable AI for Android Malware Detection: Towards Understanding Why the Models Perform So Well?" in *Proc. 2022 IEEE 33rd International Symposium on Software Dependability, trustworthiness, and consistency Engineering (ISSRE)*, Charlotte, NC, USA, Oct. 2022, pp. 169-180. DOI: http://doi.org/10.1109/ISSRE55969.2022.00026
17. M. S. Chandana Snehal, V. Nagoor, S. Rohit, R. S., S. K. Thangavel, K. Srinivasan, and P. Kapoor, "Towards Explainability Using ML and Deep Learning Models for Malware Threat Detection," *IEEE Access*, vol. 12, pp. 89441-89459, 2024.
DOI: http://doi.org/10.1109/ACCESS.2024.3419062

AUTHOR'S PROFILE

Divish Raj O holds a Bachelor of Engineering in Computer Science and Engineering from JNN College of Engineering. Shimoga, and a Master of Technology in Computer Network Engineering from RV College of Engineering. Bengaluru. My academic work includes projects on a 3D authentication system using a Rubik's Cube and on automation implementations leveraging Ansible, Kubernetes, and scripting. I have over 2 years of professional experience working with both startups and multinational corporations, during which I gained expertise in cloud automation, including devOps tools and network systems. My interests include cybersecurity, automation, and distributed system design.

Dr Vinay V. Hegde is an Associate Professor in the Department of Computer Science and Engineering at RV College of Engineering, Bengaluru. With 17 years of teaching and 13 years of research experience, his areas of interest include Natural Language Processing and Machine Learning. He has published over 80 papers in reputed journals and conferences. A CCNA-certified professional, he also serves as a training partner in the Cisco Networking Academy, as a mentor for Atal Tinkering Labs by NITI Aayog. The Government of India actively supports innovation among students across schools in Northeast India.