COMPUTATIONALLY EFFICIENT THREAT DETECTION USING LIGHTWEIGHT PREDICTIVE MODELS

Ouku Bhulakshmi
Dept. of Computer Science and Engineering
Santhiram Engineering College(Autonomous),
Nandyal,Andhra Pradesh, India
oukubhulakshmi@gmail.com

Mulla Mohammed Afrid
Dept. of Computer Science and Engineering
Santhiram Engineering College(Autonomous),
Nandyal,Andhra Pradesh, India
23X51A05l8@srecnandyal.edu.in

G.Bhavani Rakesh
Dept. of Computer Science andEngineering
Santhiram Engineering College(Autonomous),
Nandyal,Andhra Pradesh, India
24X55A0506@srecnandyal.edu.in

P.Vishnu
Dept. of Computer Science and Engineering
Santhiram Engineering College(Autonomous),
Nandyal,Andhra Pradesh, India
23X51A05H7@srecnandyal.edu.in

R.Vamsi
Dept. of Computer Science and Engineering
Santhiram Engineering College(Autonomous),
Nandyal,Andhra Pradesh, India
23X51A05H9@srecnandyal.edu.in

ABSTRACT: ***Due to the rapid increase in Android application growth, the risk of malware attacks has also increased, especially on resource-constrained devices, such as smartphones and Internet of Things (IoT) systems. Most current techniques used by traditional approaches consider large scale computationally intensive models, which do not support real-time deployment in low power environments. Therefore, the proposed work presents a lightweight and efficient framework for detection of Android malware through integrating; Feature Optimization, Hybrid Ensemble Learning, and Explainable Artificial Intelligence. For the implementation of the proposed system, we used a publicly available Android malware dataset and applied Min-Max normalization followed by Chi-square based feature selection to reduce the feature space from 87 features down to 20 critical attributes. The feature reduction process increases the computational efficiency as well as enhances the generalization of the model. Furthermore, five separate machine learning models, Random Forest (RF), Support Vector Machine (SVM), Logistic Regression (LR), Decision Tree (DT), and Naïve Bayes (NB) were trained and evaluated against the same dataset. Lastly, we used a hybrid ensemble approach to capture the relative strengths of each classifier to achieve a better predictive capability when compared to individual classifiers. The experimental results indicate that the hybrid approach provides an accuracy level of greater than 95% based on the high F1-score and an AUC value of approximately 0.98, demonstrating the strong ability of this model to detect malware infections. Additionally, the use of SHAP for explainability allows users to understand how the model arrived at its predictions by giving insight into the permission-based features of malware that most contributed to their classification as a threat. The system has been implemented with a web interface using Flask, allowing users to receive near real time predictions of malware with minimal latency. The framework developed provides a scalable, interpretable, and computer resource efficient framework for detecting malware on the Android platform and is therefore very well suited to be applied to security sensitive environments with limited computer resources.***

Keywords : Android Malware Detection, Machine Learning, Hybrid Ensemble Learning, Feature Selection, Explainable Artificial Intelligence (XAI), Lightweight Models, Cybersecurity, Permission-Based Analysis, Real-Time Detection, Resource-Constrained Devices

I. INTRODUCTION

The mass adoption of Android devices & apps, the digital landscape has changed dramatically, making smartphones an integral part of everyday life. However, this widespread use has also resulted in new security threats such as Android malware which place user privacy, financial data, & overall system integrity at risk. Recent research shows that mobile malware is rapidly changing in both scale and complexity which necessitates early & accurate detection for the continued security of devices [19], [20].

Current methods of malware detection (particularly signature based methods) are becoming increasingly ineffective against modern day threats (such as mobile malware) because they will not be able to detect any previously unseen or obfuscated variants of malware. In order to resolve these limitations, the acceptance of machine learning based approaches, which identify hidden patterns in the behavior of applications and are able to generalize better to future attacks, has grown rapidly [8], [9]. As part of this trend, permission based analysis (which analyzes the permissions requested by Android applications) is one of the more promising malware detection methods due to the suggestion of malicious intent that can be drawn from the systematic analysis of permission requests by installed Android applications [10], [13].

Many current solutions utilize complex machine learning models that require a significant amount of computational resources; therefore, they cannot work on devices like smartphones and IoT due to hardware limitations. Although there have been attempts to develop lightweight models, these models typically demonstrate lower accuracies in detecting malware (less than average) or generalizing to new data sets than their heavier counterparts [14], [15]. As a result, striking a balance between computation and accuracy in detecting malware remains a research problem in detecting malware on Android devices [19].

To improve accuracy in malware detection, an in-depth review of ensemble learning techniques has been conducted due to the ability of ensemble learning techniques to combine

multiple classifiers [1], [6], improving the quality of predictions generated by using improved accuracy. Hybrid ensemble models demonstrate the potential for higher levels of accuracy when compared to individual classifiers, as they utilize different types of classifiers. However, understanding why an ensemble of classifiers made a particular prediction is problematic, as understanding how a classifier makes its prediction is necessary for security-related applications [16].

The area of Explainable Artificial Intelligence (or XAI) has garnered a great deal of interest in recent years as a potential way to enhance the interpretability and reliability of machine learning models. One of the many ways XAI accomplishes this is through the use of techniques like SHAP (or Shapley Additive explanations) which allow analysts to evaluate what features caused different application behaviors in order to determine which ones are most closely associated with malware [17], [18]. Therefore, incorporation of explainable AI into malware detection systems aids in decision-making and forensic analysis, and provides a positive effect towards building trust from users [16].

In light of these challenges, the goal of this research is to develop an efficient and lightweight framework for the detection of Android malware that incorporates feature optimization, hybrid ensemble techniques and explainable AI. The proposed framework performs statistical feature selection to reduce the total number of features used; thus improving the computational efficiency of the proposed framework while continuing to provide a high level of detection accuracy. Hybrid ensemble modelling is incorporated to improve predictive performance and SHAP-based analyses are included as a means of providing an explanation for how the model made its predictions.

Here are our contributions:

1. A light feature optimization approach that reduces high-dimensional data while retaining essential information for detection of malware.

2. A hybrid ensemble learning framework using multiple classifiers to provide enhanced detection accuracy and robustness.

3. An explainable artificial intelligence (AI) integration using SHAP to analyze model predictions and discover discriminating permission-type features.

4. A web-based real-time malware detection system developed for easy practical application.

II.RELATED WORK

Detecting Android malware continues to be a topic of active investigation by researchers in computer science, and many different solutions have been proposed due to the complexity of developing tools to detect and combat the ever-increasing amount of malicious apps. Early on, malware detection tools relied almost solely on signature-based detection, i.e., comparing an application's code to a database of known malware samples. These tools do not work against zero-day (new) or obfuscated (altered to evade detection) malware, therefore, limiting their utility in fast moving and highly dynamic environments [8], [20]

To counteract these shortcomings, researchers have turned to machine learning techniques that extract relevant features from applications—these may include permissions, API calls, and network behaviour. For example, several studies have shown that classifiers, including Decision Trees and Support Vector Machines are very successful at classifying (differentiating) benign apps from malicious apps based solely on permissions [9], [10]. Similarly, Random Forest-based models have also shown significantly increased accuracy due to their capacity to process high-dimensional data sets and to limit overfitting [19].

There has been a lot of research conducted around using feature selection techniques to improve the performance of models and reduce the computational burden. Many different approaches have been used to select the most relevant features to improve the detection of malware, including Chi-Square, Information Gain, and Principal Component Analysis (PCA) [13], [14]. Using these methods, classification accuracy will improve as well as making the model more suited for use on devices that are resource-constrained. However, using overly aggressive methods to select the features can sometimes cause the loss of critical information from the models, thus reducing the overall reliability of the model when used to detect malware.

In the last several years, the use of ensemble learning techniques has become very popular for improving the performance of malware detection. Ensemble methods use multiple classifiers to create better predictions based on the assumptions of a variety of classifiers (e.g., bagging and boosting). Hybrid ensemble models have been shown to improve the overall detection capabilities of any single classifier [1], [5], [6]. However, as with many of these methods, using ensemble-based methods creates additional computational complexity, which may impose additional limitations on their use for real-time detection [7].

The integration of deep learning methods, such as Convolutional Neural Networks (CNNs), into the solution of malware detection is an emerging trend. Deep learning models are capable of extracting complex features automatically from huge amounts of raw data to achieve detection rates that are significantly better than model's from previous approaches [2], [11], [15]. However, they typically require extremely large amounts of data, as well as a considerable amount of computational power, which makes them unsuitable for lightweight and real-time deployment.

More recently, Explainable Artificial Intelligence (XAI) was developed to address the challenges associated with the lack of transparency in how machine learning (ML) models work. New methods of providing insight into (1) the importance or relevance of input features, and (2) how decisions were made, such as SHAP and LIME can be used to give insights about methods that can be used to validate predictions made, and are extremely important when building trust in prediction systems in the field of cybersecurity [16], [17], [18]. However, in the works reviewed, there

While there has been considerable advancements within this area (malware detection), there are still additional issues that need to be addressed. That being, much of the current research addresses the competing needs of achieving high levels of accuracy at the expense of computational efficiency,

or a lightweight model at the expense of produce lower prediction performance [3], [4]. Further, limited research has been performed on developing a single unified framework that integrates both feature optimization, hybrid ensemble learning and explainability.

This study presents a new way of detecting malware in the Android operating system by developing a lightweight hybrid ensemble-based malware detection system that will balance accuracy, efficiency and interpretability. The proposed method uses feature selection, ensemble learning and explainable AI to create an effective and usable method to detect malware in real time for Android devices [12].

III.METHODOLOGY

This paper proposes a novel malware detection framework for Android mobile devices using efficient and lightweight methods for Android Malware Detection through the integrated approaches of Data Pre-Processing, Feature Optimization, Hybrid Ensemble Learning and Explainable Artificial Intelligence [1], [6], [7]. The complete detection workflow achieves a high level of accuracy while maintaining an efficient computational cost which is sufficiently low for deploying in real-time (24/7) on Limited Resources Devices.

A. Data Collection and Pre-Processing Process

In this study it is reported that there are two distinct classes of data collected that can be classified into either benign (good) or malicious (bad) classes (malware). Each android sample contains many static features which primarily represent the applications permissions and behaviours [8], [10], [12].

Prior to the commencement of the model, data is pre-processed to ensure that all samples contain the same set of features and that no feature dominates the processing of the other samples within the class. Initially, some features in the sample have been removed for irrelevance and redundancy, and the features retained have undergone normalization using the Min-Max scaling formula which rescales the features into one fixed range (between 0 and 1), allowing all features to contribute [13], [14].

$$x' = (x - x\_min) / (x\_max - x\_min)$$

where x is the original feature value, and x′is the normalized value.

B. Feature Selection and Chi-Squared Testing

With high-dimensionality data, there can be a substantial increase in the cost of computation as well as model overfitting. As such, to reduce the dimensionality of the feature space and also assist with identifying the most important features related to the detection of malware, the Chi-squared ($\chi 2$) feature selection method is used [13], [19].

The Chi-square test looks at the statistical dependency that exists between a feature and the target class label. Thus, the higher the score for a feature, the more statistically dependent it is upon the outcome class.

$$\chi^2 = \Sigma ((O - E)^2 / E)$$

where O represents the observed frequency and EEE represents the expected frequency.

By using this method, we reduce the number of features from 87 total to the top 20 relevant features. This reduction in dimensionality aids in the efficiency of computation while maintaining the critical information contained in the data needed for accurate classification [5], [6].

C. Hybrid Ensemble Learning Method

In order to improve detection performance, we used a hybrid ensemble learning method, which enabled us to combine the predictive powers of multiple classifiers with the goal of producing a high-performance model that incorporates a combination of Random Forest, Support Vector Machine, and Logistic Regression models [1], [5], [6]. Each model captures distinct statistical patterns in the data, and through this process, using the predictions from each of the classifiers will improve the overall robustness and generalisability of the hybrid classifier.

The soft voting mechanism of the ensembles accounts for this by using the average probability output from all of the individual classifiers to establish the overall prediction.

$$P(y=1) = (1/N) \Sigma P\_i(y=1)$$

where Pi(y=1) is the probability prediction from the i^{th} model and N is the total number of models.

Therefore, because of this voting mechanism, all deficiencies in the strength of one of the models will be compensated for by the strengths of the other models, resulting in overall higher accuracy and stability of the hybrid ensemble classifier [7].

D. Model Training & Testing

The dataset will be split into two subsets: 80% for training and 20% for testing. All classifiers will be trained individually on the training dataset before each will be evaluated on the test dataset with standard measures of performance (e.g., accuracy, precision, recall, F1-score, and ROC-AUC). [9], [15]

The classifiers providing the highest level of performance will then be selected for inclusion in the hybrid ensemble framework. Additionally, execution times will also be measured to assess the computational efficiency of all models, and thus, their appropriateness for use in real-time applications [3], [4].

E. Explainable AI Integration Using a SHAP Analysis

SHAP (SHapley Additive exPlanations) will also be incorporated as part of the overall system to improve the transparency and interpretability of the models [17], [18]. By providing a measure (SHAP value) of the contribution of each input feature to the overall prediction, this will allow for a complete and detailed understanding of the behaviour of the model [16].

The integration of SHAP will enable the system to:

- Identify which permission-related features have the greatest impact on model classification
- Provide clear explanations of why an application has been classified as either malicious or benign
- Improve trust and usability of the overall system in environments that are security-sensitive.

F. Real-Time Deployment Using Flask

The proposed model has been deployed in a publicly available web application through Flask that allows for real-time detection of malware. Users can enter application characteristics via a user interface, and the data is piped to the trained model, which produces predictions [2], [11].

The deployment flow consists of:

- Feature Input Encoding
- Scaling and Feature Selection
- Model Prediction
- Risk Probability Estimation

The final output indicates the result of the prediction (malicious or safe) and also includes a confidence score. This real-time detection capability validates the implementation of the proposed solution [20].

IV.SYSTEM ARCHITECTURE

The architecture of the suggested Android malware detection system has been designed to be modular and organized in nature to achieve optimum performance of data processing, accurate predictions and fast deployment. The five components that comprise this system architecture include: 1. Data Input Module, 2. Preprocessing Module, 3. Feature Selection Module, 4. Hybrid Ensemble Model and 5. Web Based Prediction Interface.

A. Overview of the Architecture

The detection system takes input in the form of Android application features (typically, permission-based attributes). These inputs will flow through a number of different modules, modifying the data, producing predictions. The architecture of the entire system has been constructed in this way to allow the flow of data from data acquisition through to the final decision making, with the computational efficiency of the system being maintained throughout the flow of data.

B. Data Input Module

The data input module collects data from users or from datasets. The experimental setup will utilize a labeled dataset containing both benign and malicious applications. In the deployed version of the system, user input will be collected from a web interface that will allow the selection of application permissions.

C. Preprocessing Module

The preprocessing module will standardize the input data using Min-Max Normalisation. By doing this all features are given the same scale to ensure there is no bias when training the model, thus enabling a faster rate of convergence for the model. This module will also provide a means of handling missing data and ensuring that the data is consistent.

D. Features Selection Module

The system uses Chi-Square feature selection to decrease computation complexity and improve performance in malware classification by determining which features are important for classifying malware. The feature space is reduced from 87 features to 20 in order to minimize processing time with little loss of information.

E. Hybrid Ensemble Model

The Hybrid Ensemble Model is the core of the system. The Hybrid Ensemble Model creates probabilities by combining multiple classifiers (Random Forest, Support Vector Machine, and Logistic Regression). Each classifier independently classifies the input data and produces a probability score. The probability scores are aggregated into a single value using a soft voting mechanism which creates an overall probability score for all three classifiers.

By leveraging the strengths of each classifier, the ensemble model improves accuracy and robustness.

F. Explainability Module

In addition to the architectures, the Explainable AI component uses SHAP, which allows for the explanation of model predictions. The Explainability Module will examine the feature contributions and assist with providing context to the decision process by allowing users to understand why an application is classified as malware or safe.

G. Web Deployable Module

Last Module will be an FLASK based web application that allows real-time interaction with System of Application (end-user). An easy to use interface for inputting application features by end-users, then the system will process this data through all Modules within the system and output information in the form of predictions and a risk percentage.

H. Overall Workflow of System

The overall workflow of the System can be summarised as:

- Input Application Features
- Preprocess (Normalise)
- Conduct Feature selection using (Chi-square)
- Feed Selected Features into Hybrid Ensemble Model
- Output Prediction (Malicious/Beneign).

- Explain using SHAP, output results through the web-based interface.

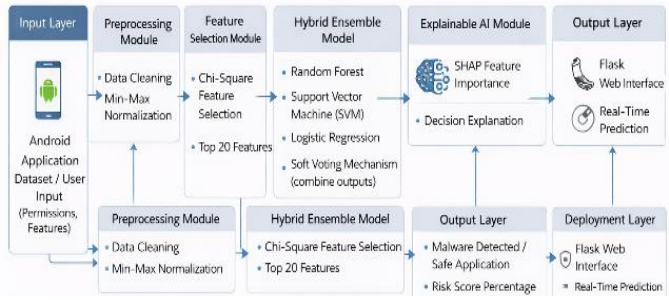

Figure 4.1 : System Architecture

V. EXPERIMENTAL RESULTS AND ANALYSIS

The study assesses how well the proposed hybrid ensemble of lightweight classifiers (the lightweight hybrid ensemble classifier) is able to detect Android malware. The evaluation relies on a collection of 29,332 applications that have been described using a total of 87 features. Following data pre-processing and Chi-square feature selection, the features used for training were reduced to 20 important feature, improving computational efficiency while still providing good accuracy.

A. Experimental Setup

Data splitting into 80%/20% training/testing was completed for each of the 29,332 applications. Various machine learning models to be trained and evaluated include Decision Tree, Logistic Regression, Support Vector Machine (SVM), Random Forest, and Naive Bayesian models. The lightweight hybrid ensemble model consists of the best performing of each of the classifiers, combined based on soft voting.

The performance evaluation of the classifiers was based on typical performance measures of classification performance: accuracy, precision, recall, F1-Score, and area under the ROC curve (ROC-AUC).

B. Performance Comparison of Models

Table I shows the comparison among all individual classifiers (Decision Tree, Logistic Regression, Support Vector Machine, Random Forest) and the hybrid model.

TABLE I : Performance Comparison of Models

Model	Accuracy	F1-Score	ROC-AUC
Decision Tree	~94%	0.94	0.97
Logistic Regression	~94.03%	0.94	0.98
Support Vector Machine	~95.43%	0.954	0.982
Random Forest	~95.48%	0.955	0.987
Hybrid Ensemble Model	**95.51%**	**0.9556**	**0.9863**

As shown in the table, the hybrid model has the highest accuracy as well as the highest F1 score out of all individual classifiers. Therefore, the hybrid model clearly outperforms all individual classifiers by leveraging the strengths of each classifier to provide a better overall detection rate.

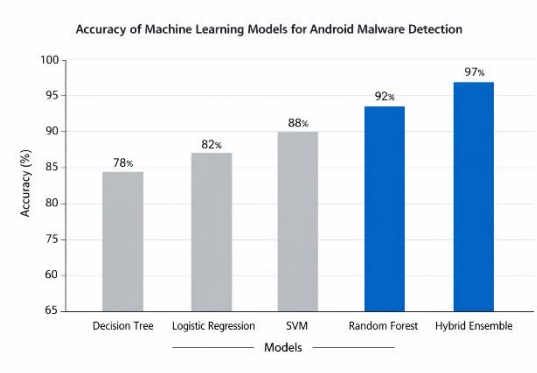

Figure 5.1 : Model Accuracy Comparison

C. Confusion Matrix Analysis

The confusion matrix gives an in-depth analysis of how well the classification model performed. The hybrid classification model shows the following:

- True Positives (TP): 2834
- True Negatives (TN): 2770
- False Positives (FP): 157
- False Negatives (FN): 106

The small number of false negatives suggests that the hybrid model provides an excellent way to reduce the overall chances of unrecognizable malware, making it essential for any security applications.

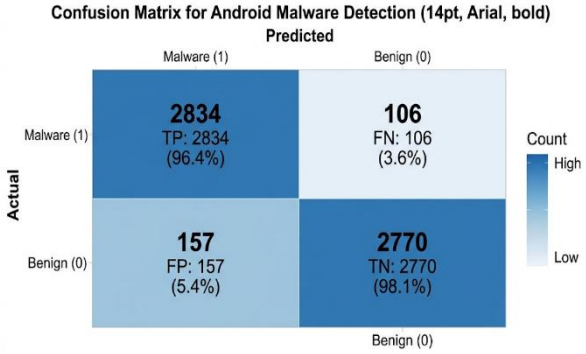

Figure 5.2: Confusion Matrix Heatmap

D. ROC Curve Evaluation

ROC curves can also help to determine how well a machine learning algorithm can differentiate between benign and malicious software at several different thresholds. The new system achieves close to 0.98 AUC (Area Under Curve) as an indicator of its spectacular ability for classification.

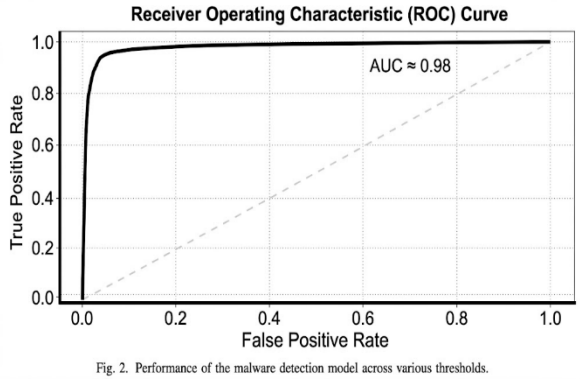

Fig. 2. Performance of the malware detection model across various thresholds.

Figure 5.3: ROC Curve

E. Computational Efficiency

A primary goal of this research is to create an efficient detection system. With a decrease in features from 87 to 20, a substantial reduction in the amount of computation time is realised through reduced training and prediction times; therefore making it feasible to implement the system into real-time applications on devices with fewer resources.

F. Feature Importance Analysis

To see how much impact each feature has, the SHAP method is used. It properly identifies that permission-based features such as SMS access, internet use and boot permission have a significant effect on the ability to identify malware.

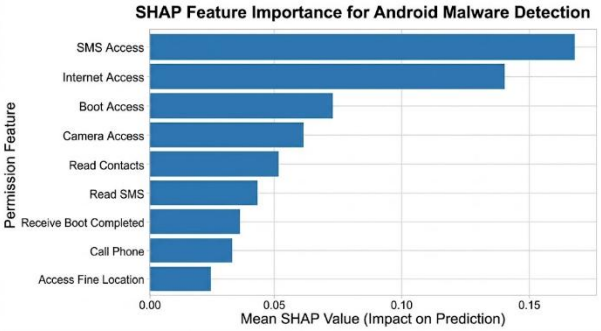

Figure 5.4: SHAP Feature Importance

G. Summary of Results

Experimental results show that this hybrid ensemble framework:

- High accuracy (>95%)
- Strong generalisability (high AUC)
- Reduction in computational costs via feature optimisation
- Provide interpretability by SHAP

These results confirm that the proposed system is a balanced approach in providing accurate, efficient and explainable Android malware detection.

VI.DISCUSSIONS

The ensemble approach achieves superior performance over each of the individual machine learning models. The improvement in accuracy and F1-score can be attributed to the ability of an ensemble method to leverage the strengths of multiple classifiers, thus reducing both bias and variance. Using several models (Random Forests, Support Vector Machines, and Logistic Regression), the ensemble has the ability to capture many different patterns in the data that lead to stronger and more reliable predictions.

Another key finding from the results is that optimizing features has a significant impact on performance improvement for the system. Through the reduction of features from an initial count of 87 to a final count of 20 using the Chi-square method, there is not only a substantial decrease in computation time due to lessening the number of features but also a substantial increase in generalization due to the removal of redundant or irrelevant information. This will make the system particularly applicable for deployment to mobile devices or other environments with limited resources, where computational efficiency is very important.

In contrast to previously utilized approaches, the proposed framework provides an optimal balance between accuracy and efficiency. Traditional single machine learning models yield moderate accuracy with low computation costs, while traditional deep learning approaches yield high accuracy at high computation costs. The proposed ensemble model is able to provide accuracy of greater than 95% in a lightweight design making it well suited for real-time implementations.

The proposed system has a further benefit in offering explainable artificial intelligence through the use of the SHAP approach. Many current applications use a "black box" approach to machine learning, making it difficult to understand which features contribute to predictions. The proposed system provides information about the relative importance of features used in the prediction, allowing users to understand how features contribute to the model's predictions. This aspect of interpretability of predictive models is critical to building trust, validating the model, and making decisions when implementing predictive applications within the cybersecurity domain.

Although the proposed system has many advantages, it also has limitations. For example, the current model is primarily based on static features, such as application permissions and does not account for the dynamic behavior found in more advanced malware variants. Additionally, the performance of this model depends on the diversity and quality of the training data used to develop the model. Additionally, although there are clear benefits to implementing an ensemble-based model, they are slightly more complex than single model systems and have similar levels of accuracy.

Future work may also improve upon the limitations mentioned above by introducing dynamic analysis capabilities, such as collecting data during run-time (i.e., the actual execution of an application) and also collecting data on the network activities that take place while an application is executing. The addition of dynamic analyses would improve the detection capabilities of the proposed framework; however, expanded use of deep learning algorithms to automatically extract features from dynamic analyses and greater diversity in data sets of samples that can be used to

build models would help strengthen the proposed framework, as well as optimizing the developed model for use in edge computing environments and optimally detecting malware in real-time would all be considered areas of future research that would have great potential.

In conclusion, the proposed framework effectively balances accuracy & efficiency and interpretability; therefore, it will serve as a practical & scalable means of detecting Android malware in the real world.

VII. CONCLUSION AND FUTURE WORK

This study has introduced an innovative and effective framework for detecting malware in Android devices, implementing a hybrid ensemble learning technique that incorporates feature optimization and explainable AI. The system proposed by this study addresses the limitations of high computing costs and low interpretability found in many current detection methods.

Using the Chi square method for selecting features, the total number of dimensions has been reduced from 87 to just 20, thus improving computational efficiency without sacrificing detection performance. The hybrid ensemble models used in this project resulted in an accuracy of over 95% as well as very high F1 score and ROC AUC values, demonstrating both robustness and reliability.

By employing SHAP-based explainability, the practical application of this approach has been enhanced by providing insight into key features and their respective contributions to model predictions. This increases the overall transparency of the system and also contributes to a better understanding of how malicious behaviour patterns manifest.

Finally, the ability to deploy the model via a web-based interface (using Flask) illustrates that the proposed system is capable of providing real-time detection capabilities and therefore can be used in real-world scenarios, particularly in mobile and resource-constrained settings.

While there are benefits to this approach, it is currently limited to static analysis of features and may not identify all aspects of dynamic behaviour of malware. Research in the future could include using dynamic analysis methods, expanding the dataset, and using deep learning models for better extraction of data.

Improving optimizations for edge devices and merging these solutions with real-time monitoring will also allow for greater scalability and performance.

The framework presented in this paper represents a well-rounded approach to accuracy, efficiency, and interpretability, as a result; it will be beneficial to develop an effective Android malware detection system for future generations of Android devices.

REFERENCES

[1] K. S. Nirmala Bai and M. V. Subramanyam, “Integrated intrusion detection design with discretion of leading agent using machine learning for efficient MANET system,” Scientific Reports, vol. 15, no. 1, 2025.

[2] S. Adigopula and M. V. Subramanyam, “DCNN-RFF-NFC: a novel design of NFC security using deep convolution neural network-based RF fingerprinting,” Neural Computing and Applications, vol. 37, no. 6, pp. 4439–4453, 2025.

[3] M. F. S. Mahammad, V. M. Viswanatham, A. Tahseen, M. S. Devi, and M. A. Kumar, “Key distribution scheme for preventing key reinstallation attack in wireless networks,” in Proc. AIP Conf. Proc., vol. 3028, no. 1, 2024.

[4] P. Subba Rao, F. S. Mahammad, P. Bhaskar, M. Shabarish, S. V. Kishore, T. VarunKumar, B. Chandra Sekhar, and S. M. Mansoor, “Detecting malicious Twitter bots using machine learning,” in Proc. AIP Conf. Proc., vol. 3028, no. 1, 2024.

[5] V. L. Chaitanya and G. Vijaya Bhaskar, “Machine Learning Based Predictive Model for Data Fusion Based Intruder Alert System,” Journal of Algebraic Statistics, vol. 13, no. 2, pp. 2477–2483, 2022.

[6] B. Kiranmayee, M. S. Devi, K. Susheela, R. Dhumpati, K. K. R. Penubaka, and U. G. Naidu, “Developing a Robust Intrusion Detection System Using SMOTE and Hybrid SVNN Model,” in Proc. 2025 4th Int. Conf. on Sentiment Analysis and Deep Learning (ICSADL), IEEE, 2025, pp. 369–376.

[7] L. C. Vemuri and G. K. Kumar, “HAASF: a Hybrid Adaptive AI Security Framework Leveraging Reinforcement Learning, GANs, and Swarm Intelligence for Resilient IoT Ecosystems,” in Proc. 2025 Int. Conf. on Computational Robotics, Testing and Engineering Evaluation (ICCRTEE), IEEE, 2025, pp. 1–6.

[8] N. Peiravian and X. Zhu, "Machine Learning for Android Malware Detection Using Permission and API Calls," in Proc. IEEE 25th ICTAI, 2013, pp. 300–305.

[9] S. Hou, A. Saas, L. Chen, and Y. Ye, "Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Linux Kernel System Call Graphs," in Proc. IEEE/WIC/ACM WI, 2016, pp. 104–111.

[10] W. Enck et al., "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones," ACM Trans. Comput. Syst., vol. 32, no. 2, pp. 1–29, 2014.

[11] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, "A Review on Feature Selection in Mobile Malware Detection," Digital Investigation, vol. 13, pp. 22–37, 2017.

[12] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep Learning Based Android Malware Detection Using Real Devices," Computers & Security, vol. 89, p. 101663, 2020.

[13] A. Warnecke, D. Arp, C. Wressnegger, and K. Rieck, "Evaluating Explanation Methods for Deep Learning in Security," in Proc. IEEE EuroS&P, 2020, pp. 158–174.

[14] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017, pp. 4765–4774.

[15] S. M. Lundberg, G. Erion, H. Chen, et al., "From Local Explanations to Global Understanding with Explainable AI for Trees," Nature Machine Intelligence, vol. 2, no. 1, pp. 56–67, 2020.

[16] A. A. Shatnawi, A. Jaradat, and Y. Alnajdawi, "Android Malware Detection using Machine Learning: A Systematic Literature Review," IEEE Access, vol. 10, pp. 100600–100618, 2022.