International Journal of Advance Research in Science and Engineering
Volume No. 13, Issue No. 07, July 2024
www.ijarse.com
IJARSE
ISSN 2319 - 8354

ENHANCING ANDROID MALWARE DETECTION USING MULTI-LAYERED STACKING AND MACHINE LEARNING TECHNIQUES

Mrs. Kiran Pachlasiya[1], Dr. Harsh Mathur[2]

[1]*Research Scholar, RNTU, Bhopal*

[2]*Asso. Prof. CSE Dept, RNTU Bhopal*

ABSTRACT

Mobile devices are extremely vulnerable to cyberattacks due to the proliferation of Android apps. Protecting user data and device integrity requires malware detection systems that are both efficient and resilient. The effectiveness of ensemble and voting algorithms, among other machine learning approaches, in improving malware detection is investigated in this paper. To enhance the handling of unknown malware and reduce the danger of misclassification, the methodology employs a two-layered stacking approach instead of a single-layer model. At its foundational level, a diversified meta data repository is created by repeatedly cross-validating the hyperparameters of classic classifiers like SVM, KNN, and Bernoulli Naive Bayes. An enhanced and more effective method for detecting Android malware is offered by the suggested model, which outperforms standalone tuning methods by a margin of 0.9%. To test how well the model works, this study uses the Drebin dataset, which has 15,240 samples with 5,662 malicious apps and 9,578 safe ones. Future work will focus on integrating deep learning techniques and real-time data streams to further improve the model's robustness and scalability.

Keywords: Hyperparameters, Malware, Machine Learning, Android, Stacking

I. INTRODUCTION

Android has become an integral part of contemporary digital engagement because to its broad app ecosystem, low price point, and adaptability. Cybercriminals, however, have taken notice of Android's popularity and are actively seeking weaknesses to attack in order to install harmful programs. Significant financial and reputational damages, disruptions to device operation, and compromises to user privacy are all consequences of malware attacks. The intricacy and quick

development of current malware are posing a growing threat to traditional malware detection methods, even if they were successful in the past. Therefore, novel methods that make use of cutting-edge computational techniques to detect and counteract malware threats are desperately required. With its capacity to sift through massive datasets in search of intricate patterns suggestive of harmful activity, machine learning (ML) has become an invaluable resource for Android malware detection. In contrast to conventional signature-based approaches, which depend on previously specified malware signatures, ML techniques allow for the creation of models that can learn from past data and identify both known and undiscovered (zero-day) threats. These methods have shown great potential in enhancing detection rates, decreasing false positives, and adjusting to novel assault pathways. However, more robust and integrative techniques are typically required since individual ML models often struggle to generalize across varied datasets or handle the complex nature of malware activity.

A promising strategy for overcoming these obstacles is the idea of ensemble learning. Model resilience and accuracy are also improved by ensemble approaches, which combine the prediction powers of several base learners. Stacking, a kind of hierarchical ensemble learning, stands out among the others. A meta-learner is used to make the final classification decision after training many base models on the same dataset. This process is called stacking. By combining the best features of many algorithms, this multi-layered strategy improves overall performance while mitigating the effects of any shortcomings.

Because there are usually many more harmless programs than harmful ones in an imbalanced dataset, malware identification becomes much more difficult. Because of this disparity, biased models may be trained on benign data yet fail miserably when applied to malicious ones. In order to rectify this, the research makes use of methods like cost-sensitive learning and the Synthetic Minority Over-sampling Technique (SMOTE). These approaches help to equalize the dataset by creating synthetic samples of the minority class or by punishing misclassifications according to their seriousness. The model's continued sensitivity and specificity in malware detection is guaranteed by these measures.

II. REVIEW OF LITERATURE

Dibos, Md Khaled Bin et al., (2022) Cell phones are among the most popular tools in today’s rapidly evolving technological landscape. The most popular operating system has been Android from the start. Because of its immense popularity, fraudsters have been targeting this

operating system with malicious programs in an effort to steal or access sensitive user data. Identifying malicious apps during installation or runtime is of the utmost importance. In this research, we provide an Android malware detection system that uses deep learning and machine learning classifiers in conjunction with static and dynamic analysis. Using data on user rights for static analysis and data on network traffic for dynamic analysis, we attempted to assess a multilayer detection procedure. With the user's rights in the AndroidManisfest.xml file, the model can identify malware before it's loaded, and with data from network traffic, it can detect malware during execution. To improve the model's accuracy and efficiency, we have used this dataset on both the machine learning and deep learning classifiers. After cleaning and pre-processing the dataset using deep Auto Encoder, these characteristics were retrieved from actual Android apps. Being a mixed multilayer model, we have achieved some impressive accuracy rates. To round out our model's decision about malware, we suggest the most accurate classifier, which will perform admirably for both static and dynamic analysis.

Nahhas, Lojain et al., (2022) because smartphones and tablets are becoming more popular, there has been a rise in assaults targeting them. One of the biggest dangers is mobile malware, which may compromise security and cost money. Malware designed to infiltrate mobile devices is expected to keep developing and spreading. The proliferation of Android has made it the preferred target of mobile malware. Android users are at serious danger due to the fast growth of malware applications, which makes static and manual analysis of harmful files challenging. This highlights the critical need for effective malware detection and categorization for Android devices. In this area, several approaches based on Convolutional Neural Networks (CNNs) have been suggested; nonetheless, more performance enhancement is possible. Here, we provide a stacking and transfer learning strategy for successfully detecting Android malware files using two popular ML models: ResNet-50 and Support Vector Machine (SVM). To train the suggested model, we convert malicious APK files to grayscale photos using the DREBIN dataset. Compared to state-of-the-art research on the DREBIN dataset, our model outperforms them on performance metrics such as accuracy (97.8%), recall (95.8%), precision (95.7%), and F1 (95.7%).

Wang, Xusheng et al., (2022) Android malware is increasing in prevalence, which is a major concern for user security and privacy due to Android's popularity as a mobile operating system. Our proposal, MFDroid, is an Android malware detection framework based on stacking ensemble learning. It aims to address the shortcomings of both classic machine learning

techniques and the single feature selection approach, specifically targeting Android malware. In this study, we combined the outputs of seven feature selection algorithms to create a new collection of features, which we then used to choose API calls, opcodes, and permissions. Next, we trained the base learner using this, and then we used logical regression as a meta-classifier to learn the implicit information from the output of the base learners and get the classification results. The examination revealed that MFDroid achieved an F1-score of 96.0%. Lastly, we distinguished between dangerous and benign apps by analyzing each feature type. Finally, this study concludes with some broad observations. Recently, permission requests from both dangerous and benign apps have been quite similar. Put simply, the training model is unable to differentiate between malicious and benign programs, even when given permission.

Zhu, Huijuan et al., (2020) Android is an excellent choice to power the Internet of Things (IoT) because to its openness and flexibility, which have made it a mainstream smartphone platform. Its usefulness and widespread use make it an easy target for harmful programs (malware). Threats to user privacy, funds, hardware, and file integrity may be posed by malware. Within this framework, SEDMDroid is created to identify Android malware using an improved stacking ensemble of deep learning algorithms, which is based on research into malware behavior. An ensemble of Multi-Layer Perception (MLP) classifiers—the method's foundational layer—and a fusion Support Vector Machine (SVM) classifier make up this two-tier architecture for stacking ensembles of deep learning. A double disturbance strategy, which involves perturbing both the sample and the feature space, is at the heart of this design; it ensures that the base classifiers are accurate and diverse, and it learns the implicit supplementary information from the trained base classifiers' outputs to optimize classification efficiency containing an average accuracy of 89.29%, the suggested technique is tested on a dataset containing multi-level static characteristics that include sensitive API, permission, monitoring system events, and permission-rate.

III. EXPERIMENTAL SETUP

In order to improve and assess the efficacy of machine learning models for malware detection, a thorough methodology is included into the system design. The approach makes use of a stacking ensemble technique with two layers, which combines several base and meta classifiers that have been optimized via grid search. Data preparation, model training, and assessment are the main parts of the system setup.

Data Preprocessing

To make the values of the independent properties more consistent throughout the dataset, normalization is used. By keeping variance values near to 1, this approach speeds up calculations, especially in the AdaBoost meta-classification. The process of normalization guarantees that features are distributed around nearby spots, which allows for more precise and rapid learning.

Model Training and Hyperparameter Tuning

In order to optimize model accuracy, the grid search method is used to fine-tune a number of machine learning algorithms. The algorithms include Logistic Regression, Random Forest, AdaBoost, Naive Bayes, and K-Nearest Neighbors (KNN).

Dataset Used

This study makes use of the DroidFusion on the Drebin-215 dataset, which is a collection of 215 attributes that stand for qualities taken from Android apps, including permissions, API call signatures, and more.

The public repository Kaggle is the source of this dataset, which contains 15,240 samples. Out of them, 5,662 are malicious apps and 9,578 are benign.

The features are organized into several groups, and you can find more information about them in the file that comes with it. This dataset is used to test how well feature selection methods work to improve Android malware detection accuracy.

IV. RESULTS AND DISCUSSION

Figure 1 shows the results of a confusion matrix used to assess the suggested two-layered stacking model's efficacy in detecting Android malware.

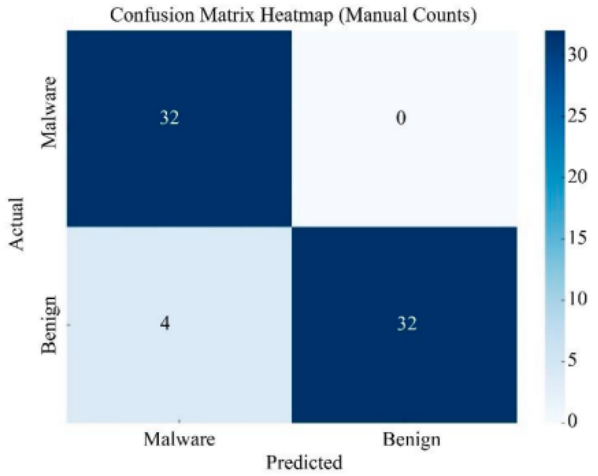

Figure 1: Confusion Matrix Heatmap for Optimized Dataset

These data are shown in the heatmap, where the actual classifications and anticipated classifications are shown on the x- and y-axes, respectively. Misclassifications are represented by the off-diagonal components (FP and FN), whilst accurate classifications are shown by the diagonal elements (TP and TN).

If we want to know how well the suggested model can differentiate between safe and dangerous apps, we need this confusion matrix. The heatmap provides a visual depiction of the model's performance, showing that it is quite accurate with few misclassifications; this proves that the two-layered stacking method is effective in detecting Android malware.

In order to make computation more manageable, the dataset used for this assessment was a limited subset, consisting of 0.452% of the original data.

Matrix of confusion: Represented in the total: 68

A comprehensive analysis of the categorization outcomes produced by the model is given by the confusion matrix. The following measures are part of it:

- True Positives (TP): Cases where malicious software was accurately detected as malicious software.
- True Negatives (TN): Cases where harmless applications were accurately classified as harmless.
- False Positives (FP): Cases where harmless apps were mistakenly labeled as malicious software.
- False Negatives (FN): When malicious software was mistakenly deemed harmless.

Hyperparameter Tuning Configuration

In order to get the best configuration, the grid search procedure requires you to select a range of values for each hyperparameter and then evaluate all potential combinations. Table 1 provides a concise overview of the tuning parameters, the optimal values achieved by this method, and the corresponding accuracy.

Table 1: Accuracy of different algorithms with tuned parameters

Algorithm	Number of Tuning Parameters	Best Values	Accuracy
Naive Bayesian	3	{'alpha': 0.01, 'binarize': 0.0, 'fit_prior': True}	85.3%

KNN	4	{'metric': 'minkowski', 'n_neighbors': 13, 'p': 1, 'weights': 'distance'}	94.12%
SVM	4	{'C': 100, 'gamma': 0.0001, 'kernel': 'sigmoid', 'probability': True}	95.37%
Random Forest	5	{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 1, 'n_estimators': 28}	94.32%
AdaBoost	3	{'estimator': DecisionTreeClassifier(), 'learning_rate': 1.02, 'n_estimators': 28}	90.26%
Logistic Regression	4	{'C': 0.1, 'dual': False, 'penalty': 'l2', 'solver': 'lbfgs'}	95.44%

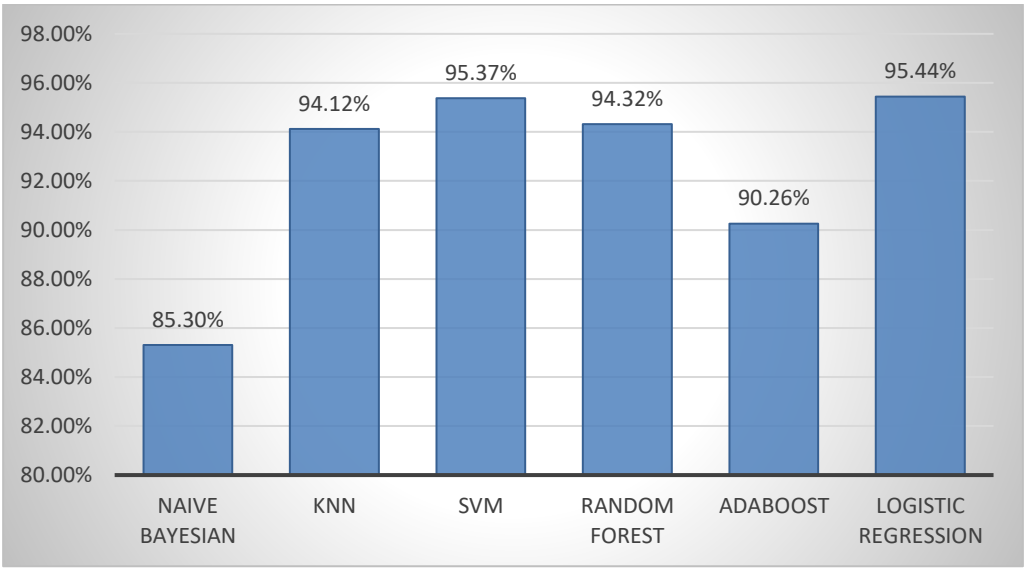

Figure 2: Accuracy of different algorithms with tuned parameters

The accuracy of different methods that were tuned using grid search to find the optimal tuning parameters is presented in Table 1. With three parameters, the Naive Bayesian method performed moderately, with an accuracy of 85.3%. With four parameters fine-tuned, the K-Nearest Neighbors (KNN) algorithm achieved a remarkable 94.12% improvement in accuracy. The Support Vector Machine (SVM), which was also fine-tuned using parameters, achieved a somewhat better accuracy of 95.37 percent. With just five tuning parameters, the Random Forest algorithm demonstrated exceptional predictive power, with an accuracy of 94.32%. An accuracy of 90.26 percent was produced by the AdaBoost classifier after optimizing it with

three parameters; this indicates competitive but marginally inferior performance in comparison to other ensemble approaches. Parameters demonstrating the Logistic Regression algorithm's efficacy in this case led to its greatest accuracy of 95.44%.

Table 2: Confusion matrix for Malware analysis

Actual	**Predicted**	**Count**
Malware	Malware (TP)	32
Malware	Benign (FN)	4
Benign	Malware (FP)	0
Benign	Benign (TN)	32
Total		68

The algorithm was able to accurately detect 89% of the real malware occurrences, as shown by the true positive rate of 0.89. The model's excellent accuracy in detecting legitimate apps is demonstrated by the zero false positive rate, which means that no harmless occurrences were incorrectly labeled as malware.

To compare the true and false positive rates across multiple threshold points, Figure 3 displays the Receiver Operating Characteristics (RoC) curve of the proposed model. Based on the confusion matrix in Table 2, the X-axis shows the false positive rate and the Y-axis shows the true positive rate.

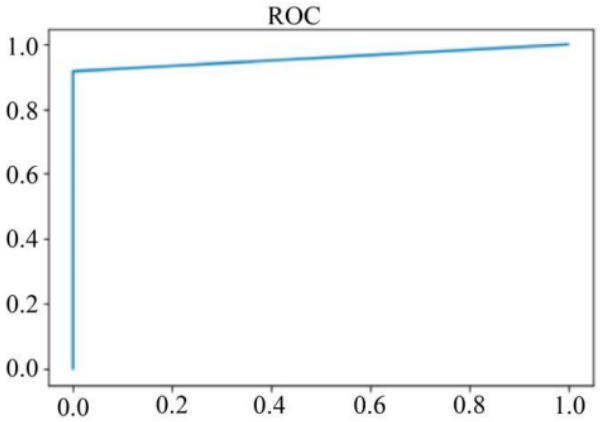

Figure 3: ROC curve analysis

V. CONCLUSION

The system obtained impressive results across a range of machine learning algorithms by making use of sophisticated data pretreatment methods, such as normalization, and by tweaking hyperparameters via grid search. After a close race, the most accurate model was Logistic Regression with 95.44% accuracy, followed closely by SVM with 95.37%. The Random Forest

and KNN classifiers demonstrated their resilience in malware detection tests with 94.32% and 94.13% accuracy, respectively. The model is a trustworthy method for differentiating between safe and harmful Android apps, as the ROC curve study shows. Improved, scalable cybersecurity solutions may be built upon this system's solid groundwork. To further improve efficiency, future research may look at using bigger datasets and more feature selection algorithms.

REFERENCES: -

[1] R. Andrea, A. I. Wahyuni, and N. F. S. Supryani, "Android-based forest fire monitoring system," *Int. J. Inf. Eng. Electron. Bus.*, vol. 14, no. 3, pp. 1–9, 2022.

[2] Md K. B. Dibos, Md Hossain, and Md Riaz, "Android malware detection system: A machine learning and deep learning-based multilayered approach," in *Proc. Springer*, pp. 277–287, 2022, doi: 10.1007/978-3-030-93247-3_28.

[3] Z. Wang, G. Li, Z. Zhuo, X. Ren, Y. Lin, and J. Gu, "A deep learning method for Android application classification using semantic features," *Secur. Commun. Netw.*, vol. 2022, no. 1, pp. 1–16, 2022.

[4] L. Nahhas, M. Albahar, A. Alammari, and A. Jurcut, "Android malware detection using ResNet-50 stacking," *Comput. Mater. Continua*, vol. 74, no. 2, 2022, doi: 10.32604/cmc.2023.028316.

[5] X. Wang, L. Zhang, K. Zhao, X. Ding, and M. Yu, "MFDroid: A stacking ensemble learning framework for Android malware detection," *Sensors*, vol. 22, no. 7, pp. 2-19, 2022, doi: 10.3390/s22072597.

[6] M. A. Aslam et al., "Breath analysis-based early gastric cancer classification from deep stacked sparse autoencoder neural network," *Sci. Rep.*, vol. 11, no. 4014, pp. 1–12, 2021.

[7] A. Taha, O. Barukab, and S. Malebary, "Fuzzy integral-based multi-classifiers ensemble for Android malware classification," *Mathematics*, vol. 9, no. 22, pp. 1–19, 2021.

[8] S. Bagui and D. Benson, "Android adware detection using machine learning," *Int. J. Cyber Res. Educ.*, vol. 3, no. 2, pp. 1–19, 2021.

[9] H. Zhu, Y. Li, R. Li, J. Li, Z. You, and H. Song, "SEDMDroid: An enhanced stacking ensemble of deep learning framework for Android malware detection," *IEEE Trans. Netw. Sci. Eng.*, vol. PP, no. 99, pp. 1–1, 2020, doi: 10.1109/TNSE.2020.2996379.

[10] R. Kumar, X. Zhang, R. U. Khan, and A. Sharif, "Research on data mining of permission-

induced risk for Android IoT devices," *Appl. Sci.*, vol. 9, no. 2, pp. 1–22, 2019.

[11] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for Android malware detection using various features," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 3, pp. 773–788, 2018.

[12] A. Naway and Y. Li, "A review on the use of deep learning in Android malware detection," *Int. J. Comput. Sci. Mobile Comput.*, vol. 7, no. 12, pp. 42–58, 2018.

[13] V. Martín, R. Rodríguez-Fernández, and D. Camacho, "CANDYMAN: Classifying Android malware families by modelling dynamic traces with Markov chains," *Eng. Appl. Artif. Intell.*, vol. 74, pp. 121–133, 2018.

[14] S. Arshad, M. A. Shah, A. Khan, and M. Ahmed, "Android malware detection & protection: A survey," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 2, pp. 463–475, 2016.

[15] P. Faruki et al., "Android security: A survey of issues, malware penetration, and defenses," *IEEE Commun. Surveys Tuts.*, vol. 17, pp. 998–1022, 2015.

[16] N. P. Bidargaddi, M. Chetty, and J. Kamruzzaman, "Combining segmental semi-Markov models with neural networks for protein secondary structure prediction," *Neurocomputing*, vol. 72, no. 16-18, pp. 3943–3950, 2009.