International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 25 Issue 03, MAR, 2025

MALWARE DETECTION IN ANDROID MOBILE PLATFORM USING MACHINE LEARNING ALGORITHMS

Mr. K. Jeevan Ratnakar[1], Jyothi Swarup Yallavula[2], Ajay Siddela[3], Pavan Sai Tadapaneni[4], Aseef Shaik[5],Sujith Paul Dasi[6]

[1]Assistant Professor, IT Department, Vasireddy Venkatadri Institute of Technology, Namburu, Guntur, Andhra Pradesh.
[23456]UG Students, IT Department, Vasireddy Venkatadri Institute of Technology, Namburu, Guntur, Andhra Pradesh –
Mail Id: jeevanratnakar@vvit.net

ABSTRACT

The increasing adoption of Android smartphones has considerably raised the potential of malware assaults, raising severe security and privacy concerns among users. Because of their reliance on preset patterns, traditional malware detection technologies such as signature-based detection and heuristic analysis frequently fail to detect new and sophisticated threats. To solve this issue, we are building an intelligent, permission-based malware detection system that leverages machine learning techniques to identify potentially hazardous programs. The fast spread of Android smartphones has transformed the digital world, providing consumers with unrivaled ease and accessibility. However, this widespread usage has resulted in an increase in security concerns, as criminal actors exploit vulnerabilities in the Android ecosystem. Traditional malware detection approaches, such as signature-based detection and heuristic analysis, fail to keep up with the constantly changing nature of malware. To solve these issues, our research provides a permission-based malware detection system that uses machine learning to detect potentially hazardous programs before they undermine device security. Our technology examines the permissions sought by Android applications in order to discover abnormalities and unusual behavior that might indicate malware. Using a dataset of both benign and malicious apps, the system is taught to detect patterns that discriminate between safe and malicious applications. We use advanced machine learning models like Artificial Neural Networks (ANN) and Support Vector Classifiers (SVC) to improve detection accuracy. The system swiftly analyzes permission data, allowing for real-time categorization of APK files with high confidence ratings, providing users with timely and trustworthy security information.

Keywords: *malware, benign, permissions, genetic algorithms, artificial neural networks, and support vector classifiers.*

1. INTRODUCTION

The fast adoption of Android smartphones has transformed mobile computing, providing users with a broad ecosystem of applications for communication, work, entertainment, and more. However, Android's enormous adoption has made it a tempting target for bad actors looking to exploit platform weaknesses. Malware attacks have gotten more complex, using innovative evasion tactics to get beyond standard protection measures. To protect user data and device integrity in today's dynamic threat landscape, effective and intelligent malware detection systems must be developed. Adware: This promotion malware application tosses undesirable promotions on the client screen and produces income.

Traditional malware detection approaches, such as signature-based methods and heuristic analysis, frequently fail to keep up with the ever-changing nature of malware. Signature-based techniques rely on preset patterns to detect malicious software, rendering them useless against zero-day assaults and fresh malware variants. While heuristic-based approaches can detect previously unknown threats, they can produce a large number of false positives or fail to catch nuanced harmful actions. To solve these constraints, researchers have turned to machine learning approaches, notably those that use program permissions to identify possible dangers.

Permission-based malware detection takes use of the fact that Android programs must disclose the permissions necessary to access system resources. Malicious apps frequently seek excessive or unneeded permissions in order to conduct unwanted operations such as data theft, spying, or remote device control. Analyzing application permission patterns allows for the proactive detection of suspicious behavior and viruses. This research investigates the use of machine learning models, especially Artificial Neural Networks (ANN) and Support Vector Classification (SVC), in combination with Genetic Algorithms (GA) for feature optimization, to improve the accuracy and efficiency of permission-based malware detection.

The primary goal of this project is to create a scalable and adaptable malware 2etectionn system that can recognize both known and developing threats. The system learns to distinguish permission-based patterns suggestive of dangerous behavior after training on a large sample of benign and malicious programs. Genetic algorithms are critical in improving the feature selection process, letting the model to focus on the most important permissions that contribute to malware categorization. This solution, implemented inside a web-based framework, allows users to submit APK files and obtain real-time safety forecasts.

This work is significant because it has the potential to provide a proactive protection mechanism for Android users, giving them a reliable tool to analyze the security risks associated with applications before installing them. By combining machine learning and evolutionary methods, this initiative advances Android security research and provides a practical solution for minimizing the rising danger of malware. The suggested approach improves detection accuracy, minimizes false positives, and adapts to the ever-changing nature of mobile security threats, resulting in a safer digital environment for all users globally.

2. LITERATURE REVIEW

We investigate existing harmful app detection systems. Nisha et al. [1] suggested detecting repackaged Android malware with mutual information and chi-square feature selection algorithms. The random forest classifier had the greatest accuracy of 91.76% among the deployed classifiers. However, Nisha et al.'s method is limited to the 88 uniquely recognized permissions for the analysis, which may be further refined to include only the damaging ones. Similarly, Sandeep [2] gathered information from the apps and carried out exploratory data analysis (EDA). The EDA strategy focuses on detecting malware during the installation process by utilizing deep learning techniques. Sandeep's detection framework used a variety of variables, including permissions, to mimic the behavior of the programs. With Random Forest as the classifier, it obtained 94.6% accuracy. The technique includes 331 characteristics for classification, which can be further tuned. Li et al. [3] proposed a permission-based detection system named SIGPID based on permission usage analysis.

Li et al. used a multilevel data trimming strategy to pick characteristics. They were able to find 22 relevant permissions by using three stages of pruning: Permission Ranking with Negative Rate (PRNR), Support Based Permission Ranking (SPR), and Permission Mining with Association Rules. Similarly, Wang et al. [4] employed a Multilevel Permission Extraction (MPE) technique, focusing on automatically recognizing the permission interaction that helps distinguish between benign and malicious programs. The dataset comprised 9736 apps from each category set—benign and malicious—were tested, and the findings showed a detection rate of 97.88%. Similarly, Sun et al. [5] employed static analysis in their study, proposing a set of program characteristics that included critical API calls and permissions, and evaluated them using the Extreme Learning Machine (ELM) approach. Sun et al. intend to identify with minimum human interaction. Sun et al. created an automated tool named WaffleDetector. The suggested tool achieved 97.14% accuracy using the ELM approach.

However, the findings are based on a tiny sample of 1049 programs from third-party or unauthorized marketplaces. Zhu et al. [6] suggested a low-cost malware detection system that extracts a collection of system events, permissions, and sensitive APIs, then calculates the permission rate based on the main attributes [7]. Zhu et al. used the ensemble Rotation Forest (RF) to create a model for classifying malicious and benign APKs [8]. The method Symmetry 2022, 14, 718 5 of 19 gave over 88% detection accuracy, over 3.3% better than the SVM classifier. However, Zhu et al.'s approach is tested on a small APK dataset [9]. Furthermore, the suggested feature set may be adjusted and tested against different machine learning classifiers to enhance detection accuracy.

The majority of the approaches used a permission-based feature set to distinguish between benign and malicious APKs. Permission-based approaches are commonly used to identify malicious behavior since they are quick and nearly always accurate. Because the examination is conducted before the App is installed, any damage to the device is avoided.

3. SYSTEM ANALYSIS

The growing prevalence of Android malware necessitates the creation of effective detection techniques capable of distinguishing between benign and malicious applications. Traditional malware detection methods, such as signature-based and heuristic approaches, frequently fail to detect new and advanced threats. To address these restrictions, machine learning approaches have been widely embraced, with a focus on permission-based analysis. The justification for this technique is that rogue programs frequently request an uncommon combination of permissions, which might indicate hazardous intent.

The system analysis for this project focuses on enhancing the malware detection process by using machine learning models to identify programs based on the permissions they seek. It detects the most important properties that help with malware detection, minimizing the entire feature set while retaining excellent accuracy. The primary objective is to enhance classification accuracy, minimize false positives, and maximize computing efficiency. The suggested methodology improves detection accuracy by improving feature selection approaches, resulting in a more efficient and effective classification process. The system evaluates many supervised Machine learning techniques, including Support Vector Machine (SVM), Random Forest, Rotation Forest, and Naïve Bayes, to find the best successful model for malware classification.

A. EXISTING SYSTEM:

The current approach, called Permission-based Detection of Malicious Applications Using Machine Learning (PerDRaML), uses a multi-level methodology to identify malware based on permission analysis. The system collects and finds critical features—primarily permissions—from a collection of Android applications, then uses several machine learning models to categorize them as benign or malignant. After considerable investigation, PerDRaML effectively finds the five most important traits that contribute to malware detection, dramatically enhancing classification accuracy.

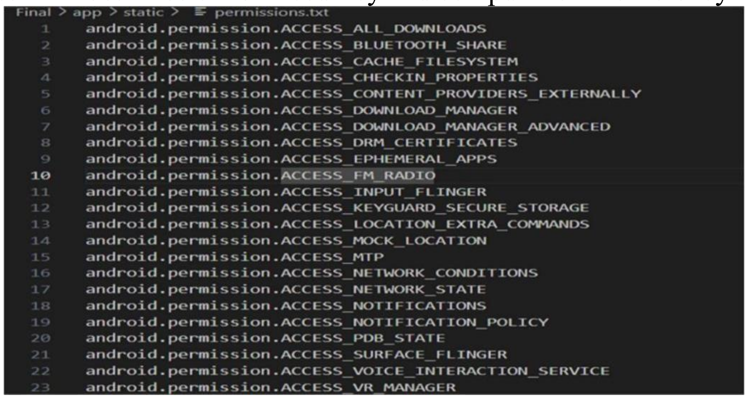
The system achieved good malware detection accuracy using Random Forest, Rotation Forest, and Naïve Bayes models (89.96%, 86.25%, and 89.52%, respectively). Furthermore, compared to previous techniques, the method optimizes up to 77% of the feature set, resulting in higher assessment metrics such as precision, sensitivity, accuracy, and F-measure. The approach reduces computational overhead while preserving detection accuracy. The present system's key components include gathering datasets of harmful and benign APKs, creating and extracting features, fine-tuning permissions, and applying supervised machine learning algorithms for classification. The fundamental goal of this strategy is to increase malware detection effectiveness while limiting the number of chosen permissions, achieving a balance of efficiency and accuracy.

B. DATASETS

The dataset utilized in this study is made up of Android applications divided into two categories: harmful and benign. These programs are assessed based on the permissions they seek, which are the major attributes used to detect malware. The permission data is organized in a binary format, with '1' indicating that an application has requested a certain permission and '0' indicating the lack of that request. The dataset, which compiles permission lists from both dangerous and benign applications, provides a solid foundation for training machine learning models to efficiently categorize applications.

The dataset, obtained from a GitHub repository, has around 400 characteristics and 1,700 individual values linked with various permissions. The permission-based method ensures that the dataset contains crucial behavioral indications for Android applications, allowing the models to learn patterns that discriminate between safe and potentially hazardous software. The dataset is labeled, with each program classed as "malware" or "benign," allowing supervised learning systems to make accurate predictions.

Users can submit an APK file, which is then evaluated for permission requests. To identify malware, the user is given the choice of using one of two machine learning models: Artificial Neural Network (ANN) or Support Vector Classifier. This option adds flexibility to the detection process, allowing users to pick the model that best meets their accuracy and computational efficiency needs.



```
Final > app > static > permissions.txt
android.permission.ACCESS_ALL_DOWNLOADS
android.permission.ACCESS_BLUETOOTH_SHARE
android.permission.ACCESS_CACHE_FILESYSTEM
android.permission.ACCESS_CHECKIN_PROPERTIES
android.permission.ACCESS_CONTENT_PROVIDERS_EXTERNALLY
android.permission.ACCESS_DOWNLOAD_MANAGER
android.permission.ACCESS_DOWNLOAD_MANAGER_ADVANCED
android.permission.ACCESS_DRM_CERTIFICATES
android.permission.ACCESS_EPHEMERAL_APPS
android.permission.ACCESS_FM_RADIO
android.permission.ACCESS_INPUT_FLINGER
android.permission.ACCESS_KEYGUARD_SECURE_STORAGE
android.permission.ACCESS_LOCATION_EXTRA_COMMANDS
android.permission.ACCESS_MOCK_LOCATION
android.permission.ACCESS_MTP
android.permission.ACCESS_NETWORK_CONDITIONS
android.permission.ACCESS_NETWORK_STATE
android.permission.ACCESS_NOTIFICATIONS
android.permission.ACCESS_NOTIFICATION_POLICY
android.permission.ACCESS_PDB_STATE
android.permission.ACCESS_SURFACE_FLINGER
android.permission.ACCESS_VOICE_INTERACTION_SERVICE
android.permission.ACCESS_VR_MANAGER
```

Figure 1. List of permissions.

This information can help users and app stores make educated judgments regarding an app's safety before it is installed or distributed. The categorization findings may also be used to apply preventative steps, such as disabling malicious applications or flagging them for further investigation, resulting in a more secure Android environment**.**

C. PROPOSED SYSTEM:

The suggested solution focuses on permission-based malware detection, assessing permission requests made by Android applications to identify potentially hazardous activity. The approach is intended to categorize submitted APK files as dangerous or benign by utilizing machine learning models trained on permission data. The development process consists of several steps, beginning with establishing project objectives, configuring the appropriate development environments, and gathering a varied dataset of Android applications, including both benign and malicious samples. To make the obtained data more usable for machine learning models, it is preprocessed with techniques such as cleaning, filtering, and feature extraction.

Feature engineering is an important part of this system, since it identifies and optimizes essential permission-based characteristics utilizing techniques such as feature selection and dimension reduction. The model selection process includes studying and implementing appropriate machine learning methods, such as Artificial Neural Networks (ANN) and Support Vector Classifiers (SVC), to ensure high accuracy, scalability, and interpretability. To improve its efficacy, the system is rigorously trained and evaluated using common performance criteria like as accuracy, precision, and recall. Once trained, the models are incorporated into a real-time detection system that lets users to input APK files for examination.

To guarantee dependability, thorough testing is performed on a variety of benign and malicious applications, including previously unreported samples, with the detection system validated against ground truth labels. The system is also geared for efficiency, scalability, and interoperability with Android devices. Finally, extensive documentation is created to outline the project's workflow, methodology, and outcomes. The system is intended not just to help end users make educated judgments regarding program safety, but also to function as a strong security solution that can be incorporated into app stores for proactive malware identification.

D. PROPOSED METHODOLOGY

The suggested approach for detecting Android malware is permission-based, with machine learning models used to identify applications as benign or dangerous. The process starts with project planning, identifying goals, and creating the proper development environment. A broad dataset of Android applications, including both benign and malicious instances, is compiled, and metadata and permission-based characteristics are retrieved. The dataset is pre-processed into binary format, with each permission represented as either present (1) or missing (0), resulting in a feature set for model training. Feature engineering techniques like as selection and dimensionality reduction are used to enhance the dataset, ensuring that the detection models work optimally.

For classification, two machine learning models are used: Artificial Neural Networks (ANN) and Support Vector Classifiers (SVC). To ensure dependability, these models are trained and assessed against a variety of performance criteria such as accuracy, precision, and recall. The learned models are incorporated into a detection system, which allows users to upload an APK file for examination. The system then processes the permissions sought by the application and classifies them accordingly. The result contains a classification label (malicious or benign), a confidence score showing the accuracy of the prediction, and other program metadata such as SDK version, name, and file size.

To improve usability and security, the system is tested against a number of apps, including known and undiscovered malware samples, to ensure its efficacy. The final version is tailored for realtime analysis and may be implemented into application stores to identify malware before it spreads. Comprehensive documentation is also given, including the methodology, system workflow, and deployment processes to ensure simplicity of use and future upgrades. The goal of this technique is to provide a fast, scalable, and accurate solution for detecting Android malware using permission-based categorization.

E. ALGORITHMS USED

The proposed Android malware detection model uses a mix of Support Vector Classifiers (SVC), and Artificial Neural Networks (ANN) to improve the accuracy and efficiency of detecting dangerous applications based on permission requests. Each method is important at different phases of the detection process, helping to create robust and scalable malware categorization systems.

1. SUPPORT VECTOR CLASSIFIER

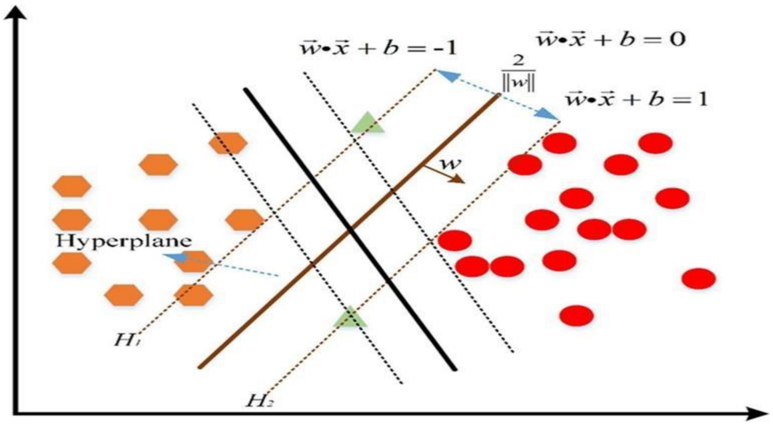

$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=0$
$\frac{2}{\|w\|}$
$\vec{w}\cdot\vec{x}+b=1$
w
Hyperplane
H_1
H_2

Figure 2. Illustrates about hyperplane and margin

The Support Vector Classifier (SVC) is a supervised machine learning technique that classifies apps based on their authorization characteristics. It works by establishing an optimum hyperplane that divides benign and malicious apps with the greatest margin available. Using the kernel method, SVC effectively maps non-linearly separable data into a higher-dimensional space, allowing it to find complicated patterns in permission-based datasets. Regularization settings allow the model to strike a balance between classification accuracy and generalization, making it suited for dealing with a wide range of malware threats.

2. ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANN) are at the heart of the classification model, utilizing deep learning techniques to detect detailed patterns in Android permission data. The ANN consists of numerous layers, including input, hidden, and output layers, as well as activation functions such as ReLU and sigmoid to induce nonlinearity. The model is trained in a sequential manner, with each layer processing incoming data, extracting significant features, and improving classification accuracy. ANN regularly updates its weights using backpropagation and optimization methods to reduce classification mistakes and improve detection skills.

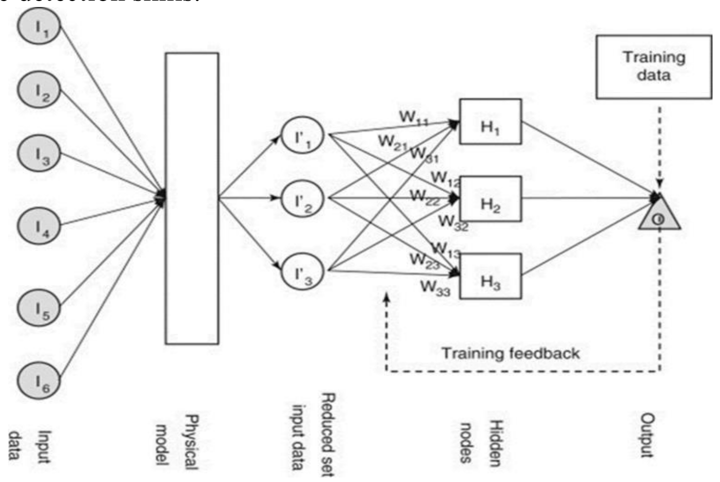


Figure 3. Artificial Neural Networks

4. IMPLEMENTATION

Importing Necessary Packages: The implementation imports NumPy, Pandas, scikit-learn, Keras, and Androguard. The script defines functions for data loading, feature selection using a Genetic Algorithm (GA), training and evaluating models (SVM and ANN), and integrating a Flask web interface for APK classification.

Extracting Permissions from Input File: The function extracts permissions from an APK file and returns a binary feature vector. It initializes a dictionary to store feature values, iterates over a predefined feature list, and assigns binary values based on the presence or absence of permissions. The final vector is converted into a NumPy array.

Loading Dataset: The dataset is loaded from a CSV file into a Pandas DataFrame. The target variable (class) is separated from features, and Label Encoder is used to convert class labels into numerical values. An SVC model is instantiated, and 5-fold cross-validation is applied to evaluate the model's mean squared error before feature selection.

Optimizing Best Feature Set using Genetic Algorithm: A custom Genetic Algorithm-based feature selection method is implemented. The GeneticSelector class initializes a population of feature subsets, evaluates fitness scores using cross-validation, performs selection, crossover, and mutation, and iterates for multiple generations to obtain the optimal feature subset. The best feature set is then used for training models.

SVC as Estimator: Genetic feature selection is performed using Support Vector Classifier (SVC). The GeneticSelector is trained with a population size of 200 over seven generations. The best feature subset is selected, and fitness scores are plotted. The trained selector is saved as a ga.pkl file.

Splitting Data and Training ANN: A neural network model is defined using Keras. The model consists of multiple dense layers with 256, 128, and 32 neurons, using dropout regularization (0.2) between layers. The model is compiled with stochastic gradient descent (SGD) and binary cross-entropy loss. Training occurs over 175 epochs with a batch size of 32. The trained model is saved as ANN_GA.pkl.

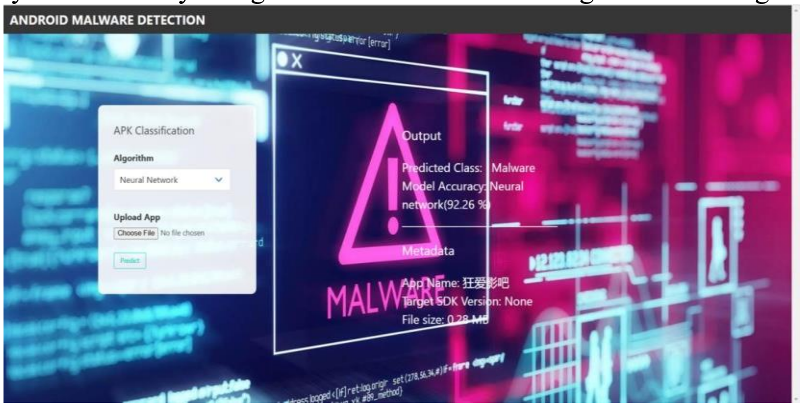
Splitting Data and Training SVC: A train-test split is performed, and GridSearchCV is used to finetune SVC hyperparameters. The best parameters and classification report are obtained. The trained SVC model is saved as svc_ga.pkl.

Loading Pickle File: A CustomUnpickler class handles loading pickled objects. The classify function extracts metadata from an APK file and creates a permission feature vector. Based on the chosen classifier (ANN or SVC), the function predicts whether the app is Benign or Malware. ANN uses a probability threshold, whereas SVC uses a classification label.

Interface using Flask: A Flask-based web interface is developed for users to upload APK files for classification. The backend loads the trained ANN and SVC models, processes input files, and displays classification results along with app metadata (name, SDK version, size). The application runs in debug mode.

5. RESULTS

The findings of this experiment demonstrate the efficacy of the created Android malware detection system. The user interface, created with Flask and HTML, allows users to upload APK files and select either a Neural Network or a Support Vector Classifier (SVC) for classification. When a malicious APK is uploaded, the system correctly recognizes it as malware. A benign APK is categorized as safe.



ANDROID MALWARE DETECTION

APK Classification

Algorithm: Neural Network

Upload App: Choose File No file chosen

Predict

Output

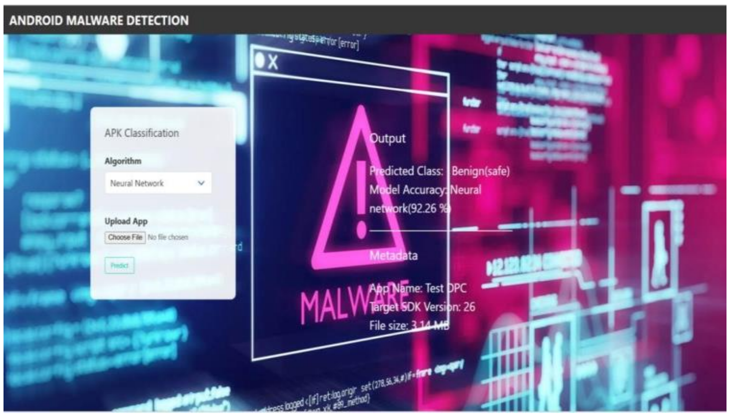
Predicted Class: Malware
Model Accuracy: Neural network(92.26 %)

Metadata

App Name: 狂爱影吧
Target SDK Version: None
File size: 0.28 MB

Figure 6. Result when malware APK is input.

The comparison of algorithm performance shows that the Neural Network model achieves a higher accuracy of 92.26%, outperforming the SVC model's 89%. These findings demonstrate the system's reliability in distinguishing between benign and malicious applications, making it an important tool for improving Android security.



Output

Predicted Class: Benign(safe)
Model Accuracy: Neural network(92.26 %)

Metadata

App Name: Test DPC
Target SDK Version: 26
File size: 3.14 MB

Figure 7. Result when benign APK is input

Table 1. Comparing Accuracy of Algorithms

Serial No	Algorithm	Accuracy
1.	Neural Network	92.26%
2.	Support Vector Classifier	89%

The performance of the Android malware detection system was evaluated using two machine learning models as shown in Table 1: a Neural Network and a Support Vector Classifier (SVC). The Neural Network model achieved an accuracy of 92.26%, making it the most effective in detecting malicious applications based on app permissions. This high accuracy is due to the model's ability to learn complex patterns and relationships in the data, allowing it to distinguish between benign and malicious applications effectively. The model was further optimized using a Genetic Algorithm (GA) for feature selection, which helped in reducing irrelevant features and enhancing classification performance. On the other hand, the Support Vector Classifier (SVC) achieved an accuracy of 89%, which is slightly lower than the Neural Network but still effective in detecting malware. The SVC model benefits from its ability to handle high-dimensional data efficiently and find the optimal hyperplane for classification. However, its lower accuracy compared to the Neural Network suggests that deep learning techniques provide better generalization for malware detection. Overall, the results indicate that the Neural Network model, when combined with Genetic Algorithm-based feature selection, is more robust and reliable for identifying Android malware.

6. CONCLUSION

The study successfully illustrates the capabilities of permission-based detection to identify Android malware using machine learning approaches. By assessing program permissions and refining feature selection with Genetic Algorithms, the system improves malware classification accuracy and dependability. The combination of Neural Networks and Support Vector Classification (SVC) enables a thorough study of application behavior, enabling accurate separation between benign and dangerous programs. The use of Genetic Algorithms for feature selection refines the model by ensuring that only the most relevant and discriminative characteristics are used, improving efficiency and lowering computational overhead. Overall, the research demonstrates the possibility of integrating cutting-edge algorithms and optimization approaches to produce a powerful malware detection solution for Android devices. This technique not only improves security, but also helps to tackle growing threats in the mobile environment.

REFERENCES

[1] Jannath, N.O.S.; Bhanu, S.M.S. Detection of repackaged Android applications based on Apps Permissions. In Proceedings of the 2018 4th International Conference on Recent Advances in Information Technology (RAIT), Dhanbad, India, 15–17 March 2018; pp. 1–8.
[2] Sandeep, H.R. Static analysis of android malware detection using deep learning.
[3] Li, J.; Sun, L.; Yan, Q.; Li, Z.; Srisa-An, W.; Ye, H. Significant Permission Identification for MachineLearning-Based Android Malware Detection. IEEE Trans. Ind. Inform. 2018, 14, 3216–3225.

[4] Wang, Z.; Li, K.; Hu, Y.; Fukuda, A.; Kong, W. Multilevel permission extraction in android applications for malware detection. In Proceedings of the 2019 International Conference on Computer, Information and Telecommunication Systems (CITS), Beijing, China, 28–31 August 2019; pp. 1–5.

[5] Sun, Y.; Xie, Y.; Qiu, Z.; Pan, Y.; Weng, J.; Guo, S. Detecting android malware based on extreme learning machine. In Proceedings of the 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Orlando, FL, USA, 6–10 November 2017; pp. 47–53.

[6] Zhu, H.-J.; You, Z.-H.; Zhu, Z.-X.; Shi, W.-L.; Chen, X.; Cheng, L. DroidDet: Effective and robust detection of android malware using static analysis along with rotation forest model. Neurocomputing 2018, 272, 638–646.

[7] Rawat, Nishant & Singh, Avjeet & Amrita. (2023). Permission-Based Malware Detection in Android Using Machine Learning. Ymer. 22(2023). 1505-1517. 10.37896/YMER22.03/C4.

[8] Ehsan A, Catal C, Mishra A. Detecting Malware by Analyzing App Permissions on Android Platform: A Systematic Literature Review. Sensors (Basel). 2022 Oct 18;22(20):7928. doi: 10.3390/s22207928. PMID: 36298282; PMCID: PMC9609682.

[9] Kshirsagar, D., & Agrawal, P. (2022). A study of feature selection methods for android malware detection. Journal of Information and Optimization Sciences, 43(8), 2111–2120. https://doi.org/10.1080/02522667.2022.2133218.