IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁑ Peer-reviewed & Refereed journal ⁑ Vol. 14, Issue 5, May 2025

DOI: 10.17148/IJARCCE.2025.14543

Android malware detection from APK file using Machine Learning

Dhanushree R[1], Doddam Sri Sai Chaithanya[2], Hemashree M R[3], Lakshmi Priya R B[4], Prof. Veeresh K M[5]

Dept of Information Science and Engineering, SJB Institute of Technology Bangalore, India[1-4]

Information Science and Engineering SJB Institute of Technology Bangalore, India[5]

Abstract: The growing number of malicious applications on the Android platform has raised significant security concerns, necessitating the development of effective detection mechanisms. This project focuses on categorizing Android applications based on their potential malicious behavior using machine learning techniques, specifically Artificial Neural Networks (ANN) and Support Vector Machines (SVM). A dataset consisting of 410 unique permission combinations, sourced from Kaggle, serves as the foundation for identifying key features that distinguish between benign and harmful applications.

The project employs two classification models: ANN, which leverages deep learning to analyze complex permission patterns, and SVM, a traditional yet highly effective machine learning algorithm known for its precise decision boundaries. Additionally, the system includes a real-time malware detection web application built using Flask. Users can upload APK files, after which the system extracts permission-based features and applies the trained models to determine whether the application is benign or malicious. By integrating ANN and SVM, this project highlights the effectiveness of permission-based machine learning models in Android malware detection. The proposed approach strengthens mobile cybersecurity by demonstrating how advanced machine learning techniques can be utilized to combat modern security threats.

Keywords: Android security, malware detection, machine learning, Artificial Neural Networks (ANN), Support Vector Machines (SVM), APK permissions, deep learning, cybersecurity, Flask web application, mobile threat analysis.

I. INTRODUCTION

The rapid growth of Andriod Applications has brought users a new level of convenience and function, but with this comes an ever-increasing array of Security concerns. Among these security threats, malicious appplications (malware) have emerged as a growing source of concern because they invade user privacy, steal sensitive information, bring systems to a stand still, and result in financial loss. Signature- Based malware detection never really seemed to garner much attention before now, as traditional approaches could never quite catch up to the ever-evolving methodologies of malware developers. Such techniques are incapable of detecting newer, hybrid kinds of malware-articulated as zero-day attacks-calling for the adoption of more flexible and robust detection solutions.

Machine learning has been emerging as a powerful and effective tool for analyzing the patterns and assessing behaviors in Android apps. Compared to traditional methods, machine-learning techniques can learn from data and, thus, adapt to new threats with more accurate and scalable malware detection. This work deliberately studies the application of machine learning algorithms: Artificial Neural Network and Support Vector Machine for the classification of Android apps into benign and malicious. The classification is performed based on application permissions requested during installation, which highly indicate application behavior and the associated security risks.

This dataset contains the characteristics of 410 unique combinations of permissions extracted from a dataset on Kaggle. The features of this dataset are used for further training and testing machine learning models. ANN is applied since it can adapt to complex nonlinear relationships within the data, while SVM, due to its capability of establishing sharp decision boundaries, is used in binary classification tasks. The main goal of this project is to create a machine learning-based malware detection system that effectively uses both ANN and SVM. The solution includes a user-friendly, Flask-based web interface that delivers real-time malware detection. Users can upload APK files via the web interface, after which the system extracts the relevant permissions features and applies the trained machine learning models to conclude whether the application is benign or malicious. The results, then presented understandably, offer users quick insight into their application's safety.

This project aims at overcoming the drawbacks of traditional malware detection and proposes an adaptive solution which is completely data-centric and makes mobile security much more effective. Besides the machine learning models, the project will also be focusing on feature-wise analysis based on permissions, model evaluation from the perspectives of accuracy and F-score, and perhaps even deployment into a real-time detection system. The scope of work spans mobile security with potential applications in cybersecurity, mobile app development, and digital forensics.

In addition, the project builds a solid base for future expansion and improvement. Future work may include integration of dynamic analysis techniques, larger and increasingly diverse datasets, and enhanced efficiency and flexibility of detection. Such upgrades will make the malware detection system stronger and more adaptable to the rising challenges of mobile security.

II. SYTSEM REQUIREMENTS SPECIFICATION

A. Software Requirements

The Android malware detection system requires compatibility with major operating systems, including Windows 10/11, macOS, and Linux. The project is developed in Python, specifically version 3.7 or later, with Python 3.8 recommended for optimal performance. It includes, among others, Flask for the development of the web interface and subsequent interaction for APK uploader and real-time prediction of malware; TensorFlow/Keras to implement and train the ANN model; and Scikit-learn to build an SVM model and utilize metrics for its evaluation, such as accuracy and the F1 score. Permissions from APK files are extracted using practices and tools such as Androguard or APKUtils, data preprocessing and manipulation are done via NumPy and Pandas, and performance and analytical results are visualized with Matplotlib and Seaborn. During development and testing, popular IDEs like Visual Studio Code, PyCharm, and Jupiter Notebook are used. In addition, tools such as APK Tool and Androguard are important for analyzing APK files and extracting useful information regarding permissions, which become core components in the feature-wise analysis of this malware detection system.

B. Hardware Requirements

For the Android malware detection system, a minimum processor specification of at least an equivalent of Intel Core i5 is required, although an Intel Core i7 or higher is recommended for faster model training and efficient APK analysis. The system should have a minimum of 8 GB of RAM, with 16 GB or more to handle large datasets and support machine learning processes better. There must be at least 50 GB of available disk space, while an SSD with 100 GB or more free space is recommended for better performance, allowing using a smoother storage and retrieval of datasets, APK files, machine learning models, and the Flask web application. Although an optional GPU might not have that much importance, it's better to have one when working with large datasets or when one has to train complex deep learning models such as ANNs. In this case, an NVIDIA GPU with CUDA support, some examples being NVIDIA GTX 1060 or better, is suggested with a minimum of 6 GB VRAM although significantly faster model training and computation can be achieved with 8 GB VRAM or higher.

C. Functional and Non-Functional Requirements

The proposed Android malware detection system primarily focuses on processing datasets by extracting permission data from APK files and utilizing machine learning models for classification. The system supports classification using Artificial Neural Networks (ANN) as well as Support Vector Machines (SVM) to separate benign from malicious applications. Performance evaluation metrics would involve accuracy, precision, recall, and F1-score. A Key feature of this system is real-time malware detection, allowing the user to access their application through a Flask-based web interface to upload the APK file. Once the file is rendered, the system extracts permission-based features that are further analyzed and classified through trained models. Outcomes are displayed in an intelligible manner, including the confidence scores where applicable. Other functionalities include logging of predictions for further reference and storing a database for facilitating further analysis and debugging.

Non-functional requirements are essential for reliable, usable, and efficient project completion. Optimization is geared towards latency minimization, while CPU utilization is maintained at optimal levels while conducting the real-time detection processes. Scalability will allow for more than one user to invoke the upload and analysis of APK files without tampering with performance. The user experience is enhanced through an easy-to-use UI and proper error handling in case invalid or corrupt APK files are selected.
Protection against malicious uploads and secure file handling is integrated to ensure the proper security of the system. Regular model retraining utilizes progressive datasets that enhance the accuracy of detection. It is designed to work independent of the platform, guaranteeing compatibility with all operating systems and web browsers. Long-drawn testing is associated with reduced false positives and negatives and works toward effective robustness of the malware detection system. Besides, the system has been developed in adherence to legal and ethical requirements and provisions under GDPR to enhance user privacy and the security of information.

The system is built to function seamlessly across different operating systems and web browsers, ensuring wide accessibility. Reliability is a key focus, with extensive testing to minimize false positives and negatives. Legal and ethical compliance, including adherence to regulations like GDPR, is also maintained. By integrating these elements, the Android malware detection system offers a secure, efficient, and user-friendly solution for enhancing mobile security.

III. SYSTEM DESIGN

The Android malware detection system aims to create an efficient, accurate, and scalable system design for the malware detection of Android applications. The functional requirements of the system involve preprocessing, training, and real-time detection. The first thing done is the extraction of the necessary permission data from the APK file for the core of the feature set to be used in training the machine learning model. Their machine learning models use two: the Artificial Neural Network (ANN) and Support Vector Machine (SVM). Based on the permission data provided, the training exercise aims at teaching the above models on how to differentiate malware apps from normal ones. Then, the two models would be used in web-based applications developed on Flash, where users upload APK files and get their predictions on whether an application is considered malware or benign. Prediction results are recorded and stored within the system so that there is traceability of previously predicted APKs. The non-functional requirements specify that the system must be able to perform in real-time, scale up to serve multiple simultaneous uploads, and remain userfriendly. There are security features integrated in place to safeguard against any malicious uploads, while the modular and maintainable coding will make future development and debugging easy. The design also focuses on the moral issues surrounding data privacy laws such as GDPR.

A. System Workflow

The workflow of the Android malware detection system is designed to process data systematically and provide accurate real-time predictions. The process begins with data collection, where the necessary permission data is extracted from Android APK files using tools like APKTool. This data is then preprocessed to prepare it for machine learning model training. The preprocessing steps include normalization, encoding, and balancing the dataset using the SMOTE (Synthetic Minority Over-sampling Technique) method to handle class imbalance and ensure that the models are trained on a well-represented dataset. After preprocessing, the data is used to train the machine learning models—ANN and SVM. These models learn to recognize the distinctive patterns of permissions that correlate with either benign or malicious behavior. Once the models are trained, they are deployed via a Flask web application. When users upload an APK file, the system extracts the required features, feeds them into the models, and generates predictions about whether the APK is benign or malicious. The results are then displayed on the user interface, which includes detailed classification results and, if possible, the confidence scores for the predictions. The entire process is designed to be efficient, with real-time malware detection that aims to provide results in under five seconds per APK. Additionally, the system logs prediction results for future reference and debugging, providing valuable insights into the system's performance.

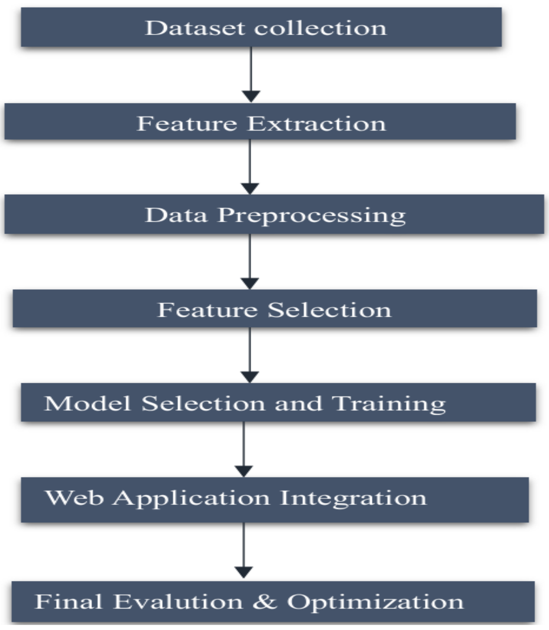

Dataset collection
Feature Extraction
Data Preprocessing
Feature Selection
Model Selection and Training
Web Application Integration
Final Evalution & Optimization

Fig 1.1: System Workflow

B. System Architecture

The android malware detection system architecture has been developed to accommodate various detection methodologies in a modular and extensible manner. This system takes input in the form of an APK file of Android and extracts permission data using APKTool. The extracted permission data forms the primary feature set for the malware detection models. The system has two different machine learning algorithms: Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs). While ANNs are able to learn multiple complicated patterns from the size of the permission data, SVMs classify the data as either malware or benign. Besides the data for APK permission, network support data will also be used for further clarification on application behavior from additional activity such as the network behavior of an application. The trained models, from which the system will function, are deployed into a web framework, such as Flask or FastAPI, to be able to process real-time APK submissions by the users. When the APK file is fed into the trained model, the application is classified accordingly and receives an output prediction result, with a confidence score where applicable, displayed to the user. The architecture is designed modularly to scale with evolving threats and allow easy integration of new machine learning models and detection techniques. It is also highly scalable, able to cater to multiple users uploading APKs simultaneously without compromising on performance. This architecture assures reliability, accuracy, and an outlook into the future of Android security threats.

The architecture of the Android malware detection system has been developed in a modular and scalable model to cater to many detection techniques. This system takes Google Android APK files as its input and uses an APKTool to extract permission data from the file. This extracted permission data constitutes the primary feature vector set for the malware detection models. The system covers two machine learning algorithms: Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs). While ANNs learn from the much-complex patterns from the size permission data, SVMs classify it as malware or benign categories. In addition to the APK permissions data, additional support was provided to use network data. This information talks about application behavior from additional activities such as the network behavior of an application, with a more concise addition of the network activity that further identifies malware. The trained models from which the system is going would be deployed with a web framework like Flask or FastAPI to process real-time APKs uploaded by users. Once the APK file is fed into a trained model, the application is classified under benign or malicious. The prediction result is final output to display along with the user, along with any confidence scores. The architecture is modular in design so that integration of new machine learning models or detection techniques is to be done, as per the evolving threat. It has also been very scalable and can accommodate multiple APK uploads without degrading performance. The architecture keeps the system definite, accurate, and adaptable to ever-changing Android security threats.

IV. IMPLEMENTATION

A. Dataset Collection

This dataset used for the project originates from Kaggle, consisting of Android APK files' permission data. Each APK in the dataset is defined by 410 unique features pertaining to permissions, where each feature denotes a permission requested by the application. Such structuring allows the machine-learning models to analyze normal and malicious applications' patterns and behaviors effectively and then accurately classify and detect malware using permission statistics.

B. Feature Extraction

Once the APK Files are collected, the next step will be to extract relevant features from each application, including, but not limited to, permissions requested by the app, API call logs, and other static features that can indicate malicious intent. Tools such as Androguard or APKTool are used for automation of this process; these will decompile the APKs and retrieve essential data points from them that are input to ML models.

C. Data Preprocessing

After the extraction of features, the raw data undergo further preprocessing to be used by machine learning. Here, the data cleaning, the handling of missing values, the normalization of features, and the encoding of categorical variables take place. Preprocessing sets a uniform standard for the dataset, making sure it is error-free and ready for further stages of feature selection and model training.

D. Feature Selection

Feature selection involves identifying and selecting the most significant and relevant features in order to improve machine learning models' performance. Such type of feature selection reduces dimensionality, removes data that is redundant or irrelevant, and boosts the speed and accuracy of the detection system. Techniques such as correlation analysis or mutual information can be used in feature ranking, where features are given a score depending on their contribution to the classification task.

E. Model Selection and Training

Once the ideal features have been selected, or optimal features, various machine learning models are chosen and trained on the preprocessed dataset-like Artificial Neural Networks and Support Vector Machines in the project-to classify APK files into harmless or poisonous samples on the basis of the extracted features. Hence, the training phase is that in which models were fully trained by iteratively feeding data through the algorithms, adjusting weights and parameters, and minimizing predictors' errors.

- Training using SVM: The SVM model would be trained using a labeled set of applications classified as benign or malicious. The hyperplane would be tuned to optimize the margin between the two classes.
- Training the ANN: Forward propagation takes place where data enter the layers, predict output, and compare it to the influenced actual output. The forward pass is followed by backward propagation in weight tuning through the error.

When a model gets an evaluation, it will use the subset test for its dataset. The goal is to identify how well it can predict a new APK whether it belongs to the benign or the malicious class. Performance metrics are taken into the consideration for an adequate measure of a model, including accuracy, precision ratio, recall ratio, and F1 score. It is also possible to apply cross-validation in supplementing the entire evaluation of model generalization.

Once evaluated and trained, the machine learning models are integrated within an Android-based application targeted for detecting malicious applications. The application uses the trained model to classify new APKs by extracting features from the APK after applying the trained model to determine if the app is benign or malicious.

The last system has been made such that it can detect malware in real time. All the apps installed by the users or the apps that get an update are analyzed automatically by Android malware detection systems that utilize trained machine learning models for analysis and categorization of the APK files. If the application is marked as malicious, the end users have the chance to be informed about the app before its installation for more security.

Users who want to interact with the malware detection system can do so through the Android app in their UIs. These UIs allow users to upload APK files, view results of detection without much interference, and even check out the detected threats for improving user experience in a more accessible and consumer-friendly manner.

F. Web Application Integration

The web application built on top of Flask integrates the models trained successfully in such a way that users can upload malware APK files for detection in real time via a friendly interface. The backend handles file processing, feature extraction, and model prediction, while the frontend presents results in a user-friendly manner.

G. Final Evaluation and Optimization

The final stage is an extensive evaluation of the entire system for accuracy, efficiency, and reliability. Optimization techniques improve the performance of the model by reducing false positives/negatives and improving the overall speed of detection. Testing and fine-tuning will continue for both the machine learning models and the web application to provide a solid and scalable solution for malware detection.

V. TESTING

Performing efficiency tests comprises a vital activity in ascertaining the capability of the modeling techniques utilized in the Android Malware Detection project. In simple terms, accuracy is the main measure through which model performance can be evaluated. This is a proportion of correct predictions made by the model.

Other measures such as precision, recall, and F1-score supplement to obtain a better assessment. Precision determines how many of the instances predicted positive are true positives and recall how many true positives have been identified correctly. The measure is the F1-score: the harmonic mean of precision and recall indicates an equitable measure of trade-off between the two measures. High precision would show few false positives, high recall would indicate that most actual positives were detected, and a high F1 would portray a balanced measure between both.

From the performance analysis of two models, which were ANN and SVM, it was proven that ANN model is superior to SVM along all the key metrics. The ANN scored 92.26% for accuracy, 94.00% for precision, 91.00% for recall, and 92.50% for F1-score against SVM's lesser values: 89.00% accuracy, 90.00% precision, 87.00% recall, and 88.50% F1-score. The performance has indicated that the ANN model was better.

DOI: 10.17148/IJARCCE.2025.14543

VI. RESULT

android.permission.SET_ALARM	android.permission.ENABLE_KEYGUARD	class
0	0	benign
0	0	benign
1	0	Malign
0	1	Malign
1	0	Malign

Fig 1.2: Dataset Last Three columns

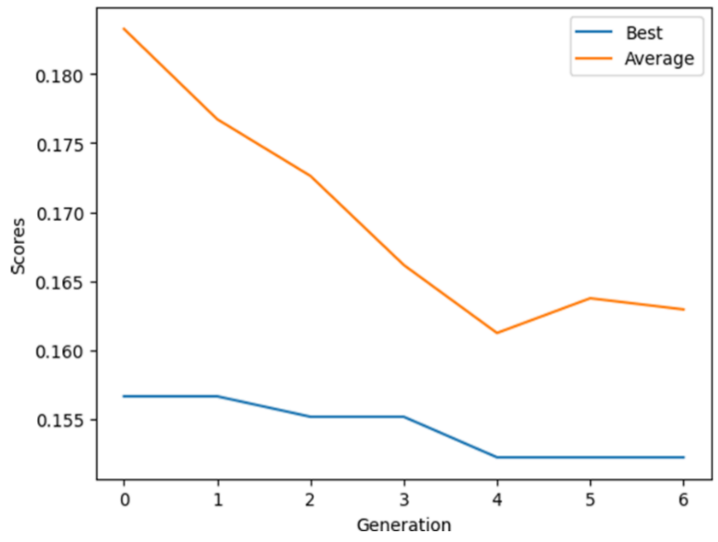
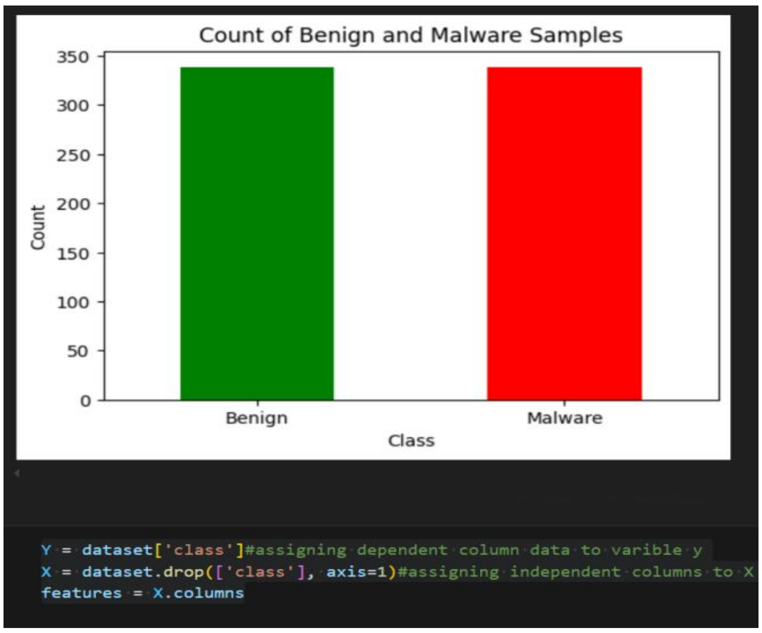


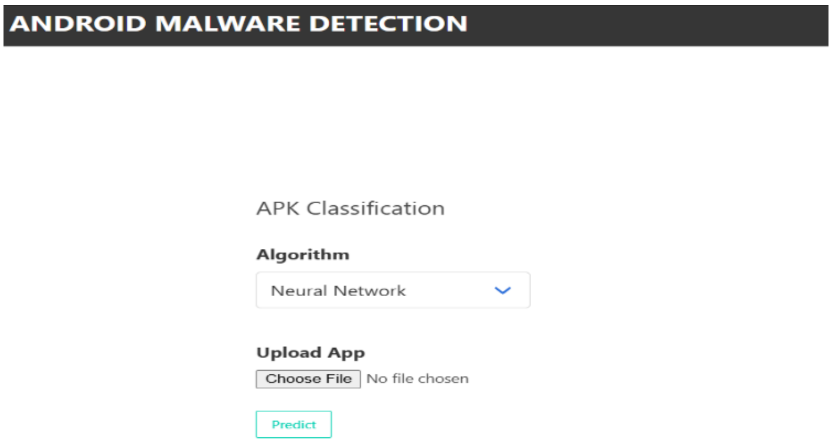
Fig 1.3: Dataset Fit and Error rate



```
Y = dataset['class']#assigning dependent column data to varible y
X = dataset.drop(['class'], axis=1)#assigning independent columns to X
features = X.columns
```

Fig 1.4: Count of Benign and Malware Samples

DOI: 10.17148/IJARCCE.2025.14543


ANDROID MALWARE DETECTION

APK Classification

Algorithm

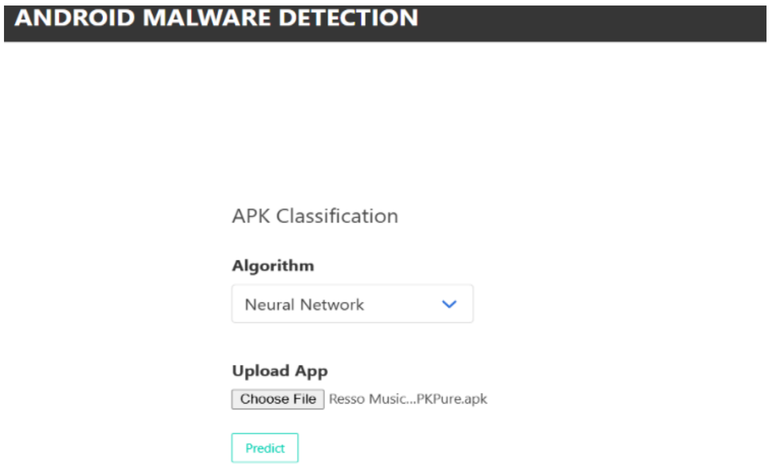
Neural Network

Upload App

Choose File No file chosen

Predict

Fig 1.5: Output web Application


ANDROID MALWARE DETECTION

APK Classification

Algorithm

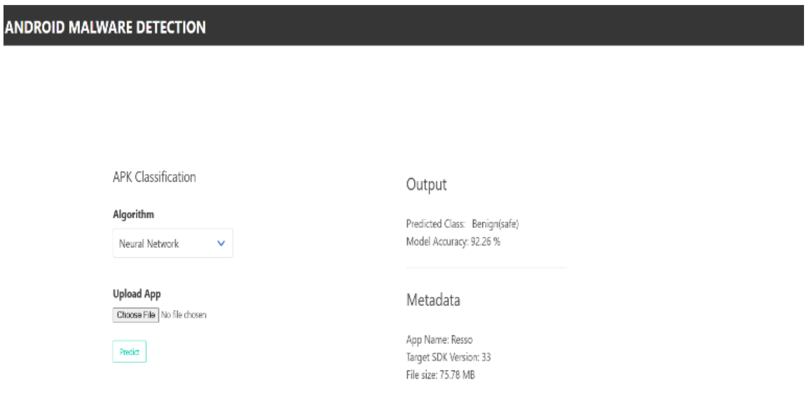
Neural Network

Upload App

Choose File Resso Music...PKPure.apk

Predict

Fig 1.5 Analyzing the apk file


ANDROID MALWARE DETECTION

APK Classification

Algorithm

Neural Network

Upload App

Choose File No file chosen

Predict

Output

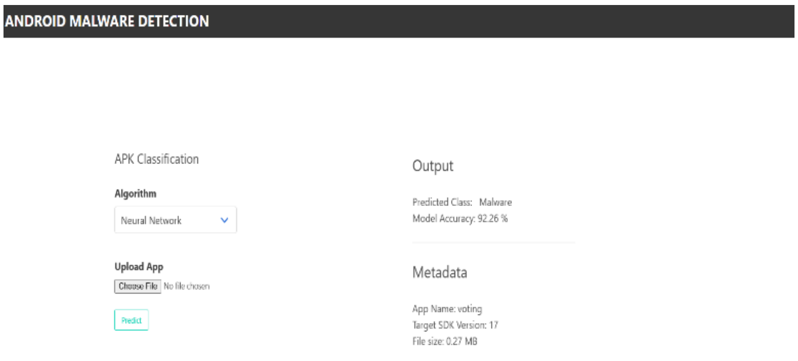
Predicted Class: Benign(safe)
Model Accuracy: 92.26 %

Metadata

App Name: Resso
Target SDK Version: 33
File size: 75.78 MB

Fig 1.6: Prediction Result of the apk file

DOI: 10.17148/IJARCCE.2025.14543


ANDROID MALWARE DETECTION

APK Classification

Algorithm: Neural Network

Upload App: No file chosen

Output

Predicted Class: Malware
Model Accuracy: 92.26 %

Metadata

App Name: voting
Target SDK Version: 17
File size: 0.27 MB

Fig 1.6: Prediction Result of the Malware apk file

VII. CONCLUSION

The Android malware or non-malware classification project demonstrates an innovative approach to tackling the growing threat of malware in mobile applications. By utilizing machine learning techniques such as Artificial Neural Networks (ANN) and Support Vector Machines (SVM), this project efficiently classifies Android APKs based on their permission patterns. The model's ability to distinguish between benign and malicious apps enhances security measures for mobile devices, providing an automated, scalable solution to malware detection.

The integration of a Flask web application allows for real-time, user-friendly interaction where users can upload APK files and receive instant classification results. This practical solution bridges the gap between complex machine learning algorithms and end-user accessibility. Additionally, with continuous improvement, model optimization, and real-world feedback, the project has the potential to evolve and adapt to new and more sophisticated malware threats.

In conclusion, this project contributes significantly to improving mobile security by offering a robust, machine-learning-driven solution that can be easily accessed and deployed, helping users safeguard their Android devices against malicious applications.

REFERENCES

[1]. A. A. R. M. R. Ahmed, S. S. Q. Rahman, and K. R. K. Sharan, "Machine Learning Techniques for Android Malware Detection: A Comprehensive Review," *IEEE Access*, vol. 8, pp. 138267–138285, 2020. [DOI: 10.1109/ACCESS.2020.3017856](https://ieeexplore.ieee.org/document/9170998)
[2]. C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. [DOI:10.1007/BF00994018](https://link.springer.com/article/10.1007/BF00994018)
[3]. M. G. N. A. Ganaie, I. A. A. Al-Quran, and A. R. Al-Bashir, "A Review of Artificial Neural Networks in Cybersecurity," *Computers & Security*, vol. 86, pp. 287–308, 2019.
[4]. G. C. D. Carbone, P. A. A. de Lima, and R. A. de Figueiredo, "A Framework for Android Malware Analysis," in *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 330–337, 2016.[DOI: 10.1109/ASONAM.2016.7752283]
[5]. M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, O'Reilly Media, 2018.
[6]. TensorFlow Team, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/ Scikit-learn Developers, "Scikit-learn: Machine Learning in Python," 2021.
[7]. C. Colaco, M. Bagwe, S. Bose, K. Jain, "DefenseDroid: A modern approach to Android Malware Detection" Strad Research 2021 doi: 10.37896/sr8.5/027
[8]. H. Zhu, Y. Li, R. Li, J. Li, Z. You and H. Song, "SEDMDroid: An Enhanced Stacking Ensemble Framework for Android Malware Detection,"in IEEE Transactions on Network Science and Engineering, vol. 8, no. 2, pp. 984-994, 1 April-June 2021, doi: 10.1109/TNSE.2020.2996379.
[9]. Mohammed K. Alzaylaee, Suleiman Y. Yerima, Sakir Sezer, DL-Droid: Deep learning based android malware detection using real devices, Computers & Security, Volume 89, 2020, 101663, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2019.101663

[10]. A. Fatima, R. Maurya, M. K. Dutta, R. Burget and J. Masek, "Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning," 2019 42nd International Conference on Telecommunications and Signal Processing (TSP), 2019, pp. 220-223, doi: 10.1109/TSP.2019.8769039.

[11]. S. Mahdavifar, A. F. Abdul Kadir, R. Fatemi, D. Alhadidi and A. A. Ghorbani, "Dynamic Android Malware Category Classification using Semi Supervised Deep Learning," 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), 2020, pp. 515-522, doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00094.

[12]. Han, J., Kamber, M., & Pei, J. (2012). Data mining: concepts and techniques. Elsevier.

[13]. G. Iadarola, F. Martinelli, F. Mercaldo and A. Santone, "Formal Methods for Android Banking Malware Analysis and Detection," 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), 2019, 10.1109/IOTSMS48152.2019.8939172.

[14]. Mohamed, Seif & Ashaf, Mostafa & Ehab, Amr & Abdalla, Omar & Metwaie, Haytham & Amer, Eslam. (2021). Detecting Malicious Android Applications Based On API calls and Permissions Using Machine learning Algorithms. 1-6. 10.1109/MIUCC52538.2021.9447594.

[15]. E. J. Alqahtani, R. Zagrouba and A. Almuhaideb, "A Survey on Android Malware Detection Techniques Using Machine Learning Algorithms.," 2019 Sixth International Conference on Software Defined Systems (SDS), 2019, pp. 110-117, doi: 10.1109/SDS.2019.8768729.

[16]. Sultana, N., Gadekallu, T. R., Naidu, S. K., & Maddikunta, P. K. R. (2019). Machine learning algorithms for malware detection: A review. Journal of Ambient Intelligence and Humanized Computing, 10(3), 1013-1029.

[17]. Datta, A., & Datta, S. (2017). Malware detection using machine learning: An overview. International Journal of Advanced Research in Computer and Communication Engineering, 6(9), 478-485.

[18]. Ayat Droos, A., Al-Attar, R., Al-Mahadeen, A., Ababneh, M., & Al-Harasis, T. (2022). Android Malware Detection Using Machine Learning. 13th International Conference on Information and Communication Systems (ICICS)