Android Malware Detection Using Extreme Gradient Boosting Algorithm

A.Meena[1], R.Karishma[2], B.Gayathri[3], K.B.Hemapriya[4]
[1]Assistant Professor, Dept of Electronics and Communication Engineering
[2, 3, 4]Dept of Electronics and Communication Engineering
[1, 2, 3, 4] K.L.N. College of Engineering, Pottapalayam, Sivagangai – 630612, Tamil Nadu, India

Abstract- *With the rise of Android devices, protecting against the surge of malicious applications has become a critical concern for device security, making Android malware detection an essential safeguard against potential security threats like malware, viruses, and hacking attempts. Traditional machine learning algorithms, such as logistic regression, have been utilized for Android malware detection. However, this algorithm may not be sufficient in detecting complex and diverse malware types.To address this issue, the utilization of the XGBoost algorithm, a renowned gradient boosting decision tree algorithm, is proposed for Android malware detection.An experiment will be conducted on a dataset comprising of more than 20,000 Android applications, containing both benign and malware samples. The efficacy of the XGBoost algorithm in accurately detecting Android malware will be demonstrated through the experiment.The proposed algorithm can assist in the early detection and prevention of Android malware, ensuring the security of Android devices is crucial to protect the privacy and data of users and to prevent any unauthorized access or malicious activity on the device.*

Keywords- XGBoost Classifier, Logistic Regression, Permission Dataset, Malware analysis.

I. INTRODUCTION

Android malware has become a significant concern for device security with the increasing use of Android devices worldwide. The complexity and diversity of malicious applications pose a significant challenge to traditional malware detection methods. Machine learning algorithms have been widely adopted in Android malware detection due to their ability to analyze large amounts of data and identify patterns.

Malware, short for malicious software, is any program or code designed to harm, disrupt, or exploit a computer system, network, or device without the user's knowledge or consent. Malware is often used by cybercriminals for various purposes, including stealing sensitive information, compromising systems, conducting espionage, or causing damage.

There are several types of malwares, including:

- Viruses: A virus is a self-replicating program that infects other files on a computer and spreads from one system to another. Viruses often attach themselves to legitimate files or programs and can cause damage to the infected system.
- Trojans: A Trojan is a type of malware that disguises itself as a legitimate program or software and tricks the user into installing it. Once installed, a Trojan can steal data, modify files, or provide unauthorized access to the system.
- Ransomware: Ransomware is a type of malware that encrypts files on a computer or network and demands payment from the victim to restore access to the data. Ransomware attacks can be devastating for individuals and organizations, and often result in significant financial losses.
- Worms: A worm is a self-replicating program that spreads from one computer to another over a network, often causing damage or disruption to the infected systems. Worms can spread quickly and can be difficult to detect and remove.
- Spyware: Spyware is a type of malware that collects information about a user's browsing habits, keystrokes, or personal data and sends it to a remote server without the user's knowledge or consent. Spyware can be used for targeted advertising or identity theft.

In Section I, the introduction for our experiment is discussed. The rest of the paper is organized as follows. Section II provides a review of related work in Android malware detection. Section III describes the dataset used in our experiment, the methodology employed for the evaluation of the proposed algorithm and the metrics used to evaluate the performance of the classifiers are given. Section IV presents the results of the experiment and compares the performance of the XGBoost algorithm with logistic regression. Section V

concludes the paper and discusses the implications of our future ideas.

II. RELATED WORKS

Logistic regression is a statistical algorithm that is commonly used for binary classification tasks, such as malware detection.logistic regression for malware detection, gather a dataset of Android applications that have been labelled as either benign or malicious. The features of each application are then extracted, such as the number of permissions requested by the application or the frequency of certain API calls. Logistic regression is trained on the dataset of labelled applications. The algorithm calculates a linear combination of the input features and applies a sigmoid function to the result to obtain a probability.

Once logistic regression has been trained, it can be used to classify new Android applications as benign or malicious. The algorithm calculates the probability of an application being benign or malicious based on its features and the learned parameters. If the probability is greater than a threshold (usually 0.5), the application is classified as malicious; otherwise, it is classified as benign.Logistic regression has several advantages for malware detection in Android applications. For example, it is a relatively simple algorithm that is easy to implement and interpret. It can handle both categorical and continuous features, making it versatile for different types of data. Logistic regression is also robust to noise and outliers in the data, making it suitable for real-world malware detection applications.However, logistic regression also has some limitations. It assumes a linear relationship between the input features and the output probability, which may not be the case in practice. Additionally, logistic regression can be sensitive to imbalanced datasets, where there are significantly more examples of one class than the other. Overall, logistic regression is an effective algorithm for detecting malware in Android applications. Its simplicity, versatility, and robustness make it a popular choice in practice. Logistic Regression are effective algorithms for malware detection in Android applications, its limitations make it less suitable for certain types of data and tasks. XGBoost, on the other hand, is a powerful and flexible algorithm that can achieve high performance in a wide range of classification tasks, making it a popular choice for malware detection in practice.

III. METHODOLOGY

This section consists of two subsections. In Section III-A, the dataset preparation and data preprocessing are discussed. The proposed classification approaches are detailed in Section III-B. The detailed flow diagram is shown in Figure 1. In Section III-C, the metrics used to measure the performance of classification algorithms are given.

A. DATA PREPROCESSING AND PREPARATION

First, we need to create a dataset of Android applications. The dataset should contain both malware and benign applications, and it should be large enough to be representative of the population of Android applications. The dataset used in this process is imported from Canadian Institute for Cybersecurity. It contains 12,999malware apps and 10,000 benign apps. The goal of data preprocessing is to prepare the data for use in the algorithm, which typically involves cleaning the data, transforming it, and selecting features.

Feature engineering involves feature extraction and feature selection. Feature extraction is the process of selecting and transforming raw data into a set of relevant features, that can be used as input for machine learning classification algorithm. Here, PCA is a technique that reduces the dimensionality of a dataset by transforming the original features into a set of orthogonal components. The components are sorted by their variance, and the top components are selected as the most important features.PCA can be used to extract the most important features from the original feature set. These features can then be used to train the XGBoost model.Split the preprocessed dataset into training and testing sets. Typically, a 70/30 or 80/20 split is used, with the larger portion being the training set.

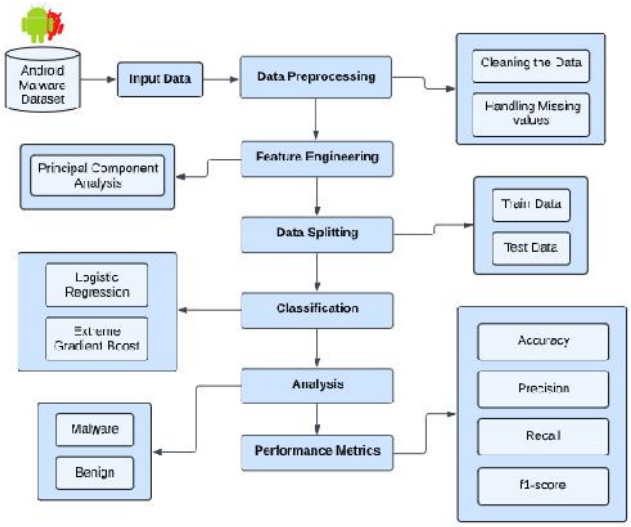

Figure 1. Flow diagram for android malware detection

B. PROPOSED CLASSIFIER

The Android operating system is vulnerable to malware attacks due to its open-source nature and the large number of apps available on the Google Play Store. Malware authors often disguise their malicious apps as legitimate ones and trick users into installing them. Traditional signature-based antivirus solutions are ineffective against such attacks. Machine learning-based malware detection techniques can be used to detect malware that has not been seen before.

XGBoost is a popular gradient boosting algorithm that has been used in various machine learning applications, including Android malware detection. It is a type of gradient boosting decision tree algorithm that can handle large datasets and capture complex relationships between input features and output classes, making it particularly useful in detecting new and previously unknown malware variants.

In the context of Android malware detection, XGBoost works by training a model on a dataset of Android applications, where each application is represented by a set of features such as permissions requested, system calls, and code structure. The model learns to classify applications as either malicious or benign based on the patterns and relationships between the features and their corresponding labels.

In Figure 2, The detailed block diagram for XGBoost Classifier is given. The XGBoost algorithm works by building an ensemble of decision trees, where each tree is trained to minimize the error of the previous tree. In each iteration, the algorithm calculates the gradient of the loss function with respect to the predicted output, and uses this gradient to update the model parameters. By iteratively improving the model, XGBoost can learn to capture complex relationships between features and malware, and can adapt to new and previously unseen malware variants.

The formula for XGBoost is as follows:

$$y = \Sigma f(x)$$

In this formula, y represents the predicted class label (e.g., malicious or benign), x represents the input features (e.g., system calls, API calls, permissions), and f(x) represents the decision function of the model, which is a combination of multiple decision trees.

The decision function is calculated as follows:

$$f(x) = wq(x) + \Sigma\, Tj(x)$$

In this equation, w is the weight of the decision tree, q(x) is the root node prediction of the decision tree, and Tj(x) represents the contribution of the jth decision tree to the prediction of the sample x. The contribution of each decision tree is a sum of the values of the leaf nodes that the sample falls into.

The weights of the decision trees and the predictions of the root nodes are learned during the training phase of the algorithm, using a gradient boosting framework that iteratively adds decision trees to the model to reduce the training error. The process starts with an initial decision tree, and at each iteration, a new decision tree is added to the model to reduce the error of the previous iteration. The weights and root node predictions of each new decision tree is learned by optimizing a differentiable loss function that measures the difference between the predicted class labels and the actual class labels of the training data.

To classify a new sample (e.g., an Android app), the XGBoost algorithm applies the decision function to the app's features and calculates the predicted probability of the app belonging to each possible class (e.g., malicious or benign).

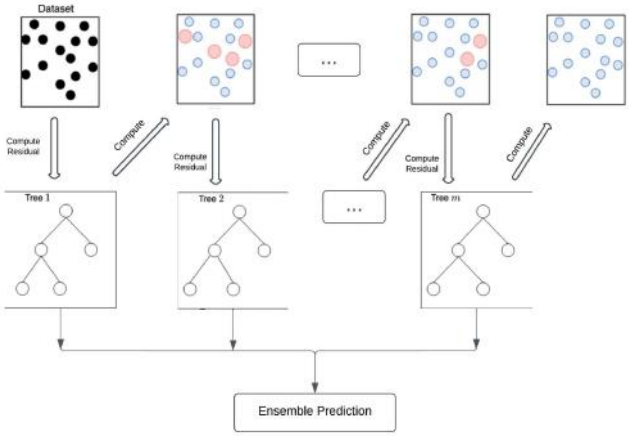

Figure 2. Block diagram for Extreme Gradient Boosting algorithm

The app is then classified as belonging to the class with the highest probability.

In addition to its ability to handle large datasets and capture complex relationships, XGBoost also provides several other benefits for Android malware detection. For example, it can handle missing values and can automatically handle feature selection, reducing the risk of overfitting and improving the generalization performance of the model.

Overall, XGBoost is a powerful machine learning algorithm for Android malware detection, and has been shown to outperform traditional approaches like logistic regression. By training XGBoost on large datasets of Android

applications and selecting appropriate features, researchers can develop accurate and efficient methods for detecting Android malware and improving the security of mobile devices.

C. PERFORMANCE MEASURE

The confusion matrix is frequently used to measure the performance of machine learning approaches. An example of a confusion matrix is shown in Table 1. Some of the information indicated in Table 1 are as follows:

TP: It is the number of samples that are actually in the ‘‘+’’ class but classified with ‘‘+’’ as a result of the classification.
TN: It is the number of samples that are actually in the ‘‘−’’ class but classified with ‘‘−’’ as a result of the classification.
FP: It is the number of samples that are actually in the ‘‘−’’ class but classified with ‘‘+’’ as a result of the classification.
FN: It is the number of samples that are actually in the ‘‘+’’ class but classified with ‘‘−’’ as a result of the classification.
By using TP, TN, FP, and FN values, accuracy in Equation 1, precision in Equation 2, and recall metrics in Equation 3 are given.

$$\text{accuracy} = \frac{TN+TP}{TN+FN+FP+TP} \quad (1)$$

$$\text{precision} = \frac{TP}{TP+FP} \quad (2)$$

$$\text{recall} = \frac{TP}{TP+FN} \quad (3)$$

Comparison with the accuracy metric may not be sufficient in experiments performed on unbalanced datasets. For this reason, it is more accurate to compare with the f-measure metric, which is the harmonic mean of precision and recall values. Equation 4 contains the mathematical representation of the f-measure metric. Considering the Table 1, two different values of precision, recall, and f-measure metrics, consisting of (+) and (−) classes, emerge. For this reason, classification algorithms are evaluated by averaging the values obtained for both classes.

$$\text{f} - \text{measure} = \frac{2*Precision*Recall}{Precision+Recall} \quad (4)$$

IV. RESULTS AND DISCUSSION

In this Section IV, the results obtained from the existing model and proposed model are detailed and interpreted.

In Android Malware Detection, XGBoost and Logistic Regression are commonly used machine learning algorithms for classifying applications as either malicious or benign. XGBoost is known for its high accuracy and scalability, making it suitable for various classification tasks, including malware detection. It is capable of handling large datasets and nonlinear relationships between features.In our experiment, XGBoost achieves an accuracy of about 95% with our permission dataset.

The results obtained from the permission dataset are discussed here. Table 2 contains the results from the Androzoo dataset. The result obtained with the Logistic Regression shows that the overall accuracy of about 90% and the precision obtained for the test data is 0.912. According to recall and f1-score obtained for Logistic Regression is 0.924 and 0.911. The result obtained with the Extreme Gradient Boost algorithm shows that the overall accuracy of about 95% and the precision obtained for the test data is 0.996. According to recall and f1-score obtained for Extreme Gradient Boost algorithm is 0.998 and 0.947.

Table 2. Results from the Androzoo dataset.

ML Model	Accuracy	f1_score	Recall	Precision
Logistic Regression	0.900	0.911	0.924	0.912
XGBoost Classifier	0.947	0.947	0.998	0.996

When compared to Logistic Regression, XGBoost outperforms it in terms of accuracy, achieving the highest accuracy score. However, Logistic Regression is known for its computational efficiency as it requires fewer computations than XGBoost. Despite this, XGBoost is also known for its computational efficiency in the context of Android Malware Detection.

XGBoost is particularly effective at capturing nonlinear relationships between features and malware, making it suitable for detecting new and previously unseen malware variants. Logistic Regression, on the other hand, may struggle to capture such relationships, as it assumes linear relationships between features and output classes.

In terms of model interpretability, Logistic Regression is a simpler model that is easy to interpret and understand, while XGBoost is a more complex model that may be harder to interpret but often achieves higher accuracy scores.

V. CONCLUSION

In conclusion, the use of XGBoost algorithm in Android malware detection has shown promising results in accurately detecting malicious applications. Overall, the XGBoost algorithm has proven to be a powerful tool in the fight against Android malware, and continued research and development in this area will undoubtedly lead to even more accurate and effective detection techniques.

Future work can include incorporating dynamic analysis, deep learning, explain-ability, adversarial attacks, and real-time detection to enhance the effectiveness, robustness and accuracy of the XGBoost algorithm.

REFERENCES

[1] Durmuş Özkan Şahin, Sedat Akleylek, And Erdal Kiliç, "LinRegDroid: Detection of Android Malware Using Multiple Linear Regression Models-Based Classifiers," Jan 2022

[2] N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, ''Machine learning aided Android malware classification,'' Comput. Electr. Eng., vol. 61, pp. 266–274, Jul. 2017.

[3] L. Wei, W. Luo, J. Weng, Y. Zhong, X. Zhang, and Z. Yan, ''Machine learning-based malicious application detection of Android,'' IEEE Access, vol. 5, pp. 25591–25601, 2017.

[4] F. Alswaina and K. Elleithy, ''Android malware permission-based multiclass classification using extremely randomized trees,'' IEEE Access, vol. 6, pp. 76217–76227, 2018.

[5] R. S. Arslan, İ. A. Doğru, and N. Barişçi, ''Permission-based malware detection system for Android using machine learning techniques,'' Int. J. Softw. Eng. Knowl. Eng., vol. 29, no. 1, pp. 43–61, Jan. 2019.

[6] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, ''A review on feature selection in mobile malware detection,'' Digit. Invest., vol. 13, pp. 22–37, Jun. 2015.

[7] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, ''Significant permission identification for machine-learning-based Android malware detection,'' IEEE Trans. Ind. Informat., vol. 14, no. 7, pp. 3216–3225, Jul. 2018.

[8] (2021). Global Market Share Held by the Leading Smartphone Operating Systems in Sales to End Users From 1st Quarter 2009 to 2nd Quarter 2018. Accessed: Oct. 30, 2021. [Online]. Available: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphoneoperating-systems/

[9] (2021). Malware Disguised as Minecraft Mods on Google Play— Kaspersky Official Blog. Accessed: Oct. 30, 2021. [Online]. Available:https://www.kaspersky.com/blog/minecraft-mod-adware-google-playrevisited/40202/

[10] H. Wang, Z. Liu, J. Liang, N. Vallina-Rodriguez, Y. Guo, L. Li, J. Tapiador, J. Cao, and G. Xu, ''Beyond Google Play: A large-scale comparative study of Chinese Android app markets,'' in Proc. Internet Meas. Conf., Oct. 2018, pp. 293–307.

[11] (2021). Mobile Malware Report—Android Malware. Accessed: Oct. 30, 2021. [Online]. Available:https://www.gdatasoftware.com/news/2019/07/35228-mobile-malware-report-no-let-up-with-android-malware

[12] A. Pektaş, M. Çavdar, and T. Acarman, ''Android malware classification by applying online machine learning,'' in Proc. Int. Symp. Comput. Inf. Sci. Cham, Switzerland: Springer, 2016, pp. 72–80.

[13] (2019). Android Malware Dataset. Accessed: Oct. 15, 2019. [Online]. Available: http://amd.arguslab.org/

[14] (2021). APKPure Android Application Store. Accessed: Oct. 30, 2021. [Online]. Available: https://apkpure.com

[15] C. Urcuqui-López and A. N. Cadavid, ''Framework for malware analysis in Android,'' Sistemas Y Telemática, vol. 14, no. 37, pp. 45–56, 2016.

[16] (2021). Access to Dataset. Accessed: Oct. 30, 2021. [Online]. Available: https://kaggle.com/xwolf12/datasetandroidpermissions