Android Malware Detection

Semester project report for Network/Internet Security, Dr. Dipankar Dasgupta instructing.

Nikhil Panda
Department of Computer Science, University of Memphis, npanda@memphis.edu
Likhitha Javvaji
Department of Computer Science, University of Memphis, ljavvaji@memphis.edu
Maniveera Venkata Sivasaikiran Parvataneni
Department of Computer Science, University of Memphis, mprvtnni@memphis.edu

ABSTRACT:
Mobile device usage has rapidly increased in recent years. By this heavy usage, the risk of cyberattacks, especially those involving malware is also increased. On Android devices, the misuse of user-granted permissions is one common method by which malware gains access to sensitive data [1]. To address this problem, researchers have developed the NATICUSdroid dataset [24], a collection of over 16,000 Android Apps with complete information on the permissions requested by each app. Researchers can create more effective methods for identifying and preventing malware problems by analyzing at this dataset in order to identify patterns and trends in the different types of permissions requested by malicious apps. In order to detect and prevent the malware they have developed effective strategies. The dataset includes information on both benign and malicious apps, so the researchers can examine the permission requests of each and identify anomalies that point the malicious behavior. The goal of our project is to detect the malware in the Android phones by using several ML algorithms [11]. According to our experimental findings, the Random Forest, Multilayer Perceptron, Gradient Boosting and Extreme Gradient Boosting models performed the best. The best recorded performance measures of above models are 97% accuracy, 97% recall, 97% precision and 97% F1-score.

1. Introduction:

Android phones are among the most popular mobile devices worldwide with a market share of more than 80%. However, because of their widespread use, they are a popular target for malware attacks. Any software intended to damage a device or system, steal data, or gain unauthorized access to a system is referred to as malware. Phishing emails, dangerous websites, and third-party software downloads are only a few of the ways malware could penetrate Android smartphones. Many applications require access to specific components of the device, like the camera or microphone, in order to function properly. However, certain apps request access to more data than they require, and they might use this access to gather sensitive data or even take complete control of the device [1]. The process of detecting malware serves to identify dangerous software and prevent it from infecting a network or computer system. Malware detection [20] is the process used to identify dangerous software and prevent it from infecting a network or computer system. Heuristic-based detection methods based on artificial intelligence are often employed nowadays to locate and eliminate malware and other forms of cyber-attacks because they may detect patterns and anomalies in data that more standard signature-based methods can ignore [3]. Additionally, heuristic-based detection algorithms can provide improved security, adaptability, accuracy, and speech quality. Heuristic-based systems have the advantage of being able to identify new and unexpected threats that may get by the classic signature-based techniques [3]. We proposed a malware detection using Machine Learning models like Logistic Regression, Random Forest, Extra Tree, Gradient Boosting, Extreme Gradient Boosting, KNN, AdaBoost, Support Vector Machine, Gaussian NB etc. [1] [2] and Neural networks along with explainability of complex model (i.e., Multilayer Perceptron) on NATICUSdroid dataset [24]. The following factors are considered since this study focuses on different ML models, such as the multi-layer perceptron and explainability (LIME, Ribeiro et al., 2016) of predictions:

- By utilizing additional ML-based classification algorithms, the NATICUSdroid dataset's potential can be increased or explored by evaluating classification quality parameters.
- Difficulty can be reduced by using LIME to break down a complex classifier's core logic [9] (i.e., what the most crucial component is).

- We can generate an explanation for a forecast by supplying the highest and lowest attributes.

1.1 Types of Malwares:

There are different types of malwares that can occur in the Android phones like SMS Trojans, Adware, Spyware, Ransomware, Banking Malware [4]. Various tools [23] are available in market to detect these know malwares.

SMS Trojans: SMS Trojans can send and receive text messages without the user's permissions, incurring expensive fees and other privacy issues for the user. SMS Trojans are often spread using phishing emails that confuse users into downloading and installing a malicious app. This can also lead to take the user's sensitive information from the device [4].
Tools: Avast Mobile Security, Bitdefender Mobile Security, etc.

Adware: Adware is a virus that can show unwanted advertisements or gather information about a user's browsing behavior. When adware is installed, it starts to show pop-up advertisements, banners, and other sorts of unwanted advertisements that are either hard to delete or difficult to remove.[4]
Tools: Malwarebytes, Ad-Aware, Mobile Anti-Virus, etc.

Spyware: Spyware is a malware that can secretly capture audio or video and access private data like call logs and text messages. Slow device performance [17], unexpected pop-up windows, modifications to device settings, and suspicious network activities are some typical warning signs of a spyware infection.
Tools: Symantec Endpoint Protection Mobile, McAfee Mobile Security, etc.

Ransomware: Ransomware is a malware that can encrypt data on the device and demands ransom in exchange for the decryption key. The most common ways that ransomware infects a victim's device are through phishing emails, malicious websites, or by taking benefit of a flaw in an operating system or piece of software. [4]
Tools: Trend Micro Mobile Security, Sophos Intercept X for Mobile, etc.

Banking Malware: Malware known as "banking malware" is capable of stealing login information and other sensitive data associated to online banking and financial services. Additionally, it can monitor SMS messages and other two-factor authentication procedures, which enables an attacker to get within security measures and access the accounts of the victim.[4]
Tools: Mobile antivirus software and Mobile threat defense (MDM) solutions, etc.

Steps to remove the malware in the Android device: [5]

i. Enter Safe Mode: Restart your device by holding the power button for few seconds.
ii. Pay Attention to the Signs of Malware on the Device: Like when we open certain apps, by that apps complete device will slow down suddenly.
iii. Uninstall Suspicious apps: Apps that asks for permissions, display advertisements etc and uninstall the apps that downloaded by itself.
iv. Use a malware detection tool to the device to remove the viruses.
v. Install and Antivirus software: Install an antivirus app from a reliable source, then do a comprehensive system scan to find and eliminate any remaining malware.
vi. Update the device: Make sure to update the device to the latest version and update all the apps when they requested to update. As a result, you will get the most recent security features and any known security vulnerabilities will be addressed.

2. Literature Review:

Numerous k-based methodologies and systems for detecting Android malware have been put out in the literature [1], as was already indicated. By studying one of the more dated Android malware datasets, such as DREBIN, Malgenome, and Contagio, or by having low detection/high false-positive rates, the majority of them employ machine learning (ML) methodologies to identify malware [5].

- **Mathur** et al. 2021 proposed **NATICUSdroid: malware detection framework [11]** to detect malware using android permissions (native and custom) as features. The solved task was binary (benign or malware) classification using ML models such as KN, SVM, DT, RF, LR, ET, XG, AB, BG along with feature importance.

- **Seraj et al. 2022** proposed **HamDroid [15]**: permission-based harmful android anti-malware detection using neural networks. The completed task includes permission-based dataset creation along with binary (benign or malicious) classification (using neural networks) which outperformed traditional ML algorithms (RF, NB, SVM, etc.)
- **Seraj et al. 2022** proposed **TrojanDroid** to **detect malware using CNN [16]**. The completed task includes creation of new dataset for trojan detection and building of CNN framework which outperformed existing traditional ML algorithms and MLP for detection.
- **Sahin et al. 2021 [17]** proposed malware detection framework using ML algorithms which utilizes concepts of **dimensionality reduction**. They reported that this concept positively influenced classification (binary) of malware.
- **Saleem et al. 2022** proposed **malware detection using feature ranking of permissions [21]**. The completed task includes building of ML model to which classify an instance either benign or malware. The feature ranking improved accuracy, recall and precision.
- **Sinan et al. 2021** proposed **AFWDroid** [22] which comprises concepts of deep **feature extraction** and **weighting scheme**. Studies showed that higher accuracy values and more competitive results in weight-sensitive classification may be attained by stat and weighting.

Dataset:

The dataset NATICUSdroid [24] is android permission-based dataset comprising 86 permissions with total instances of 29332. The target column of dataset is results (0: Benign, 1: Malware), which is two class categorical column. The dataset is slightly imbalanced (51/49) without missing values. The used dataset was gathered from two sources, including a significant amount of Android apps. which created a benign app dataset using the apps present in the Androzoo project database. More than 10 million Android apps are accessible across 18 app stores, including the Google Play Store, according to Androzoo, which also provides meta-data about them like name, size, checksum, and VirusTotal ratings.

3. Experiments and Results:

This section goes into great detail on the suggested approach taking into consideration ML/NN models, explanations, and measures. As a part of that we trained 9 machine learning models [11] and used a neural network-based model – Multilayer perceptron [7]. In the Mathur et al. 2021 paper they provided prediction using SHAP model, but we used Local Interpretable Model-Agnostic Explanations (LIME) because LIME can handle any black-box model faster with great institution considering localized parameters [9]. LIME is a model-agnostic method, which means that it may be applied to any model, independent of its design or complexity. This makes it a flexible tool for explaining the predictions of various models, such as deep neural networks, decision trees, and support vector machines. In contrast, SHAP is intended only for additive models, which restricts the range of models to which it can be used.

3.1 Data Preparation:

- **Data Gathering:** We used NATICUSdroid dataset [24] to analyze malware attacks on android apps and then we proceed to the next step of data preprocessing.
- **Data Pre-processing:** Data pre-processing involved cleaning of raw data to subset of form namely 'features', calculated number of instances and identified type of dataset to make it ready for data processing.
- **Data Processing:** Handled the missing values and removed the rows which have more than 40% of the columns as null. Identified the permissions which influence the classification categories (0 for benign and 1 for malware)
- **Feature Extraction:** In the NATICUSdroid dataset, in total we have 86 features which contains various Android app permissions. The data against each feature represents the app's need of that particular permission. Selected the top 60 most significant features based on ANOVA F-value [10].

3.2 Implementation of Models:

Below are the Machine Learning models and neural network model that are applied for the Android Malware Detection [20] [7].

A. **Logistic Regression:** A logistic regression is used to evaluate the likelihood of a binary response variable based on one or more predictor variables in the statistical technique known as logistic regression [2] [18].
B. **Random Forest:** Random Forest is an ensemble machine learning approach that combines many decision trees to increase accuracy and decrease overfitting. It accomplishes this by building many decision trees and combining their predictions to create a more reliable and precise model [2] [18].
C. **Extra Tree:** Extra Tree is an ensemble machine learning approach that combines multiple decision trees with randomized splitting thresholds to further enhance the accuracy and lower the variance of the model [2] [18].
D. **Gradient Boosting:** Gradient Boosting is an ensemble machine learning algorithm that sequentially builds several decision trees, with each subsequent tree being trained to fix the mistakes caused by the previous tree [2]. [18]
E. **Extreme Gradient Boosting (XGBoost):** Extreme Gradient Boosting (XGBoost) is a well-known machine learning technique that has been effectively applied to a wide range of problems, including regression, classification, and ranking problems. With the aim of enhancing the performance of the model, XGBoost is an expansion of the gradient boosting technique [2] [18].
F. **K-Nearest Neighbors (KNN):** KNN is a straightforward machine learning technique that predicts the response variable of a new observation by locating the k-nearest neighbors in the training data and using their majority vote [2] [18].
G. **AdaBoost:** AdaBoost is an ensemble machine learning technique that, through iteratively increasing the weights of incorrectly classified samples, combines several weak classifiers into a powerful classifier [2] [18].
H. **Support Vector Machine:** SVM is a binary classification algorithm that maximizes the margin between the two classes by locating the best hyperplane in a high-dimensional space [2] [18].
I. **Gaussian Naive Bayes:** An efficient machine learning approach for classification problems is called Gaussian Naive Bayes (GNB). Assuming a Gaussian distribution for the continuous characteristics of the input data, it is a variation of the Naive Bayes algorithm [2] [18].
J. **Multi-Layer Perceptron:** A common neural network architecture for classification is the multi-layer perceptron (MLP) [6]. It includes multiple layers of neurons, each of which computes the weighted sum of its inputs and uses an activation function to generate an output. The hidden layers carry out nonlinear transformations to extract valuable high-level characteristics for classification. The input layer gets the input information, the output layer offers class probabilities, and the hidden layers do the modifications. Back-propagation is used by MLP to train the model [6], changing the weights and biases of the neurons based on the discrepancy between the expected and actual class labels. Gradient descent is used in this procedure to determine the gradient of the loss function regarding each neuron's weights and biases. By learning non-linear decision boundaries through hidden layers, MLP is capable of handling non-linearly separable data [6]. MLP generalizes well to new data when trained with enough amounts of data and regularization strategies to avoid overfitting [6]. It can be challenging to train and optimize the model's architecture and hyper-parameters, and doing so takes a sizeable amount of data and computational resources.

As a part of our implementation of models the experimental results (accuracy, precision, recall and F1-score) of all classifiers including MLP reported in table I. We also computed Area Under the Curve for all classifiers, which is presented in figure 1.

- **Accuracy:** Accuracy measures the percentage of correctly classified instances out of all the instances in the dataset. It is calculated as: Accuracy = (True Positives + True Negatives) / (True Positives + False Positives + True Negatives + False Negatives) [25].
- **Precision:** Precision measures the percentage of correctly classified positive instances out of all the instances that were classified as positive by the model. It is calculated as: Precision = True Positives / (True Positives + False Positives) [25].
- **Recall:** Recall measures the percentage of correctly classified positive instances out of all the positive instances in the dataset. It is calculated as: Recall = True Positives / (True Positives + False Negatives) [25].
- **F1-score:** F1-score is the harmonic mean of precision and recall, and it provides a single metric to evaluate the model's performance. It is calculated as: F1-score = 2 * (Precision * Recall) / (Precision + Recall) [25].
- **Area Under Curve (AUC):** The ability of a model to discern between favorable and unfavorable cases is measured by AUC. An accurate classification of positive and negative examples as positive and negative,

respectively, is demonstrated by a model with a high AUC [8]. A model with a low AUC, on the other hand, may be producing predictions that are inaccurate and is having trouble differentiating the two classes.

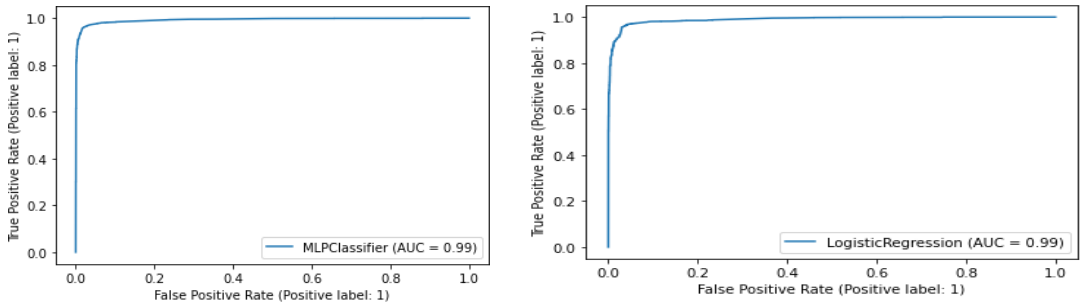
- Table I, results are without utilizing concepts of feature scaling. Feature scaling is a preprocessing technique used to standardize the range of features or variables in a dataset. It is performed to ensure that all features contribute equally to the analysis, avoiding the dominance of some features over others, which may lead to biased results.

- Table II reports best performing models with feature scaling.

[7]

Model	Accuracy %	Precision %	Recall %	F1-score %
Logistic Regression	96	96	96	96
Random Forest	97	97	97	97
Extra Tree	92	92	92	92
Gradient Boosting	97	97	97	97
Extreme Gradient Boosting	97	97	97	97
decision	96	96	96	96
Adaboost	96	96	96	96
Support Vector Machine	96	96	96	96
Gaussian NB	60	77	60	53
Multi-layer Perceptron	97	97	97	97

Table I: Accuracy, Precision, Recall and F1-score of Machine Learning Models without feature scaling



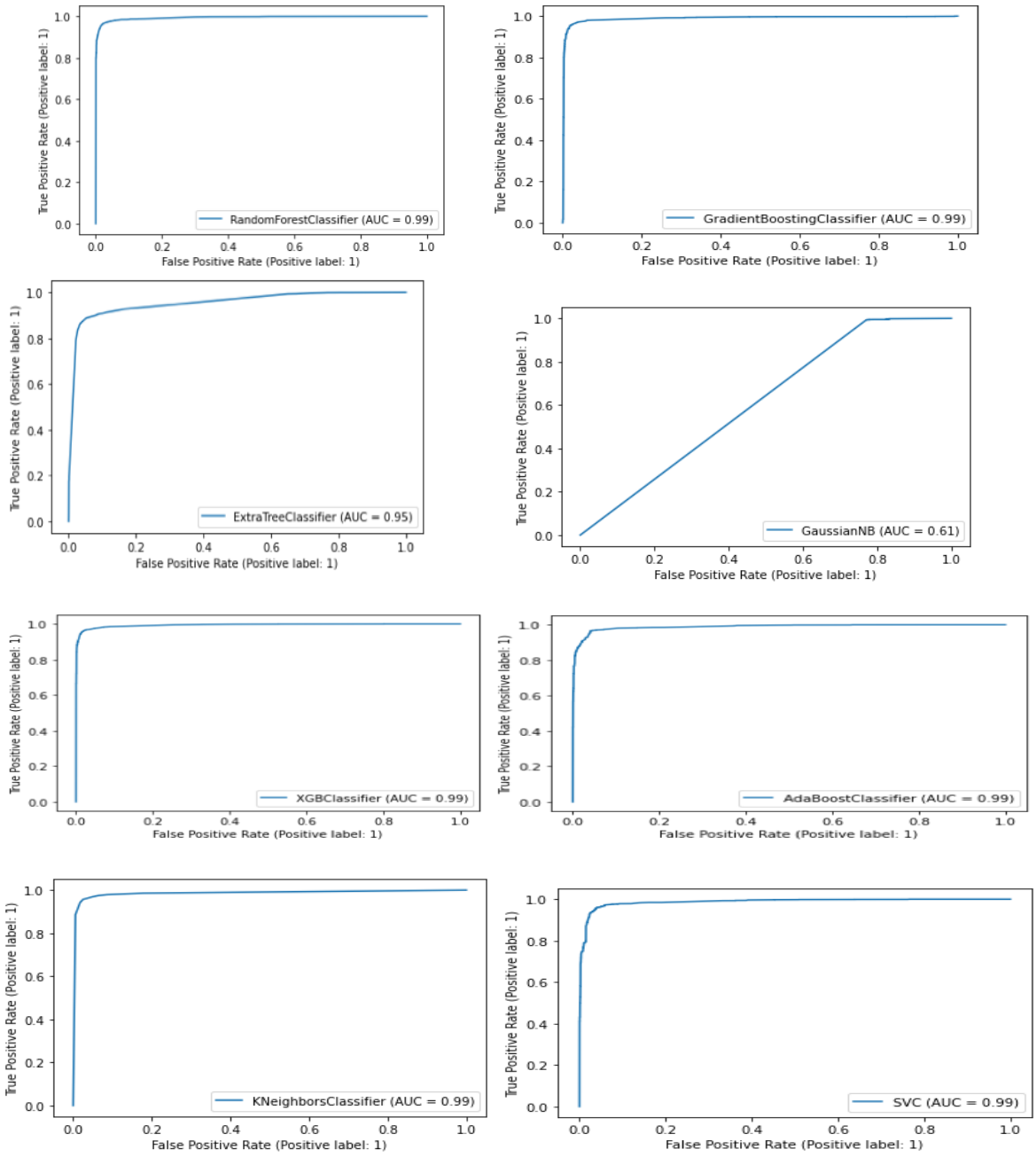


Figure 1: AUC of different classifiers [8]

Model	Accuracy %	Precision %	Recall %	F1-score %
Random Forest	97	97	97	97
Multi-layer Perceptron	**97**	**97**	**97**	**97**
Gradient Boosting	97	97	97	97
Extreme Gradient Boosting	97	97	97	97

Table II: Accuracy, Precision, Recall and F1-score of Machine Learning Models with feature scaling

From the results of Table II, we are considering MLP as preferred choice by considering below reasons.

- In general, MLPs are easier to understand than tree-based models like Random Forest and Gradient Boosting [18].
- Large datasets can be used to train MLPs, and they scale well as the amount of data increases.
- MLPs are less likely than tree-based models to overfit, especially when the number of features is high when compared to the number of observations and compared to tree-based models, MLPs require less tuning of their hyperparameters, making them potentially simpler to train and optimize.

3.3 LIME Prediction:

LIME (Local Interpretable Model-Agnostic Explanations) is a well-known technique for illuminating the forecasts of powerful machine learning models [9]. For each prediction, LIME offers a way to generate a local, instance-level explanation, which may assist in improving the model's predictability and transparency [9]. LIME's advantage is that it offers a flexible and adaptable technique to create explanations that are adapted to the particular requirements of the application and the users. For non-expert users, LIME can also be used to produce textual explanations that explain the prediction's logic in straightforward terms [9]. LIME operates by creating a group of interpretable models that closely approach the complicated model's predictions in close proximity of a given instance or observation. Choosing a specific instance for which we want a complex model's prediction to be explained is the first step in using LIME [9]. LIME next creates a collection of perturbed instances by selecting a random sample of the original instance's surroundings and perturbing the characteristics in various ways [9]. For the purpose of training the interpretable model, the perturbations are made to produce a diverse range of cases [20]. On the perturbed dataset, an interpretable model is then trained, such as a logistic regression, Decision Tree. The significance of each feature for the black box model's prediction is then described using the weights of the interpretable model. A visual or written description of the forecast can be created using these feature significance scores. In our experiment we have taken few testing instances for prediction explanations, one such example is presented in figure 2. The true label is 0 (benign) and predicted label is 0 too. The supporting features for prediction highlighted with blue colors. For this maximum 20 features are selected.

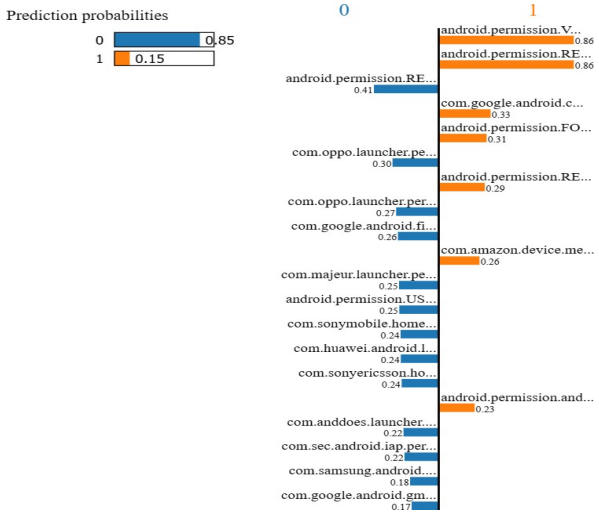
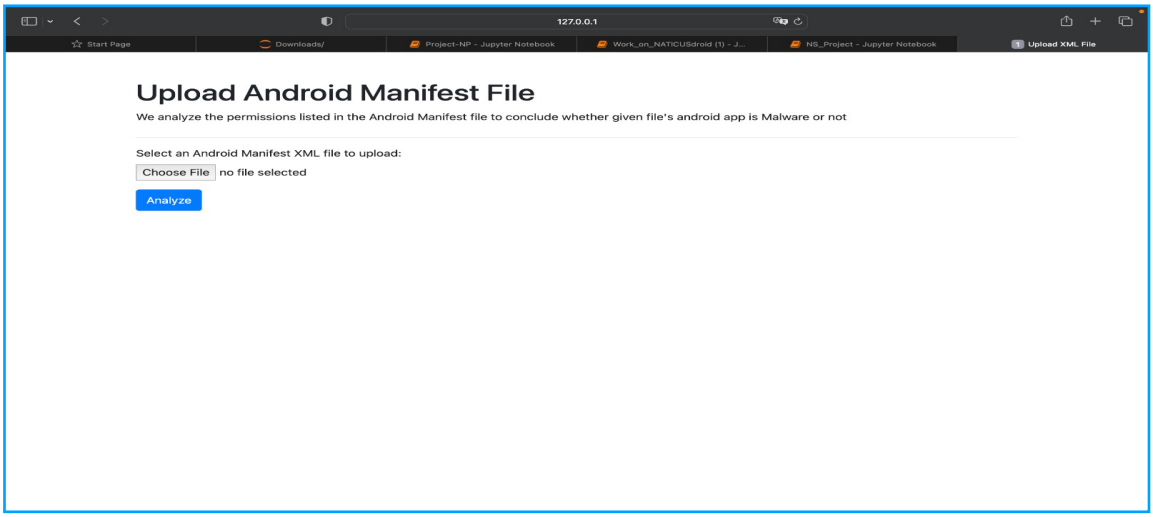


Figure 2: LIME Prediction explanation

3.4 Service Development:

Once we have a model in place to make predictions, any end-user should be consuming that model with a viable medium like an easily accessible Application/API endpoint. For this project, we built small web application which will accept the Android manifest file and perform XML processing to extract all the permissions that are required by that particular android application. These permissions will be fed into model and then trained model will predict whether given instance of android application is Malware or not.



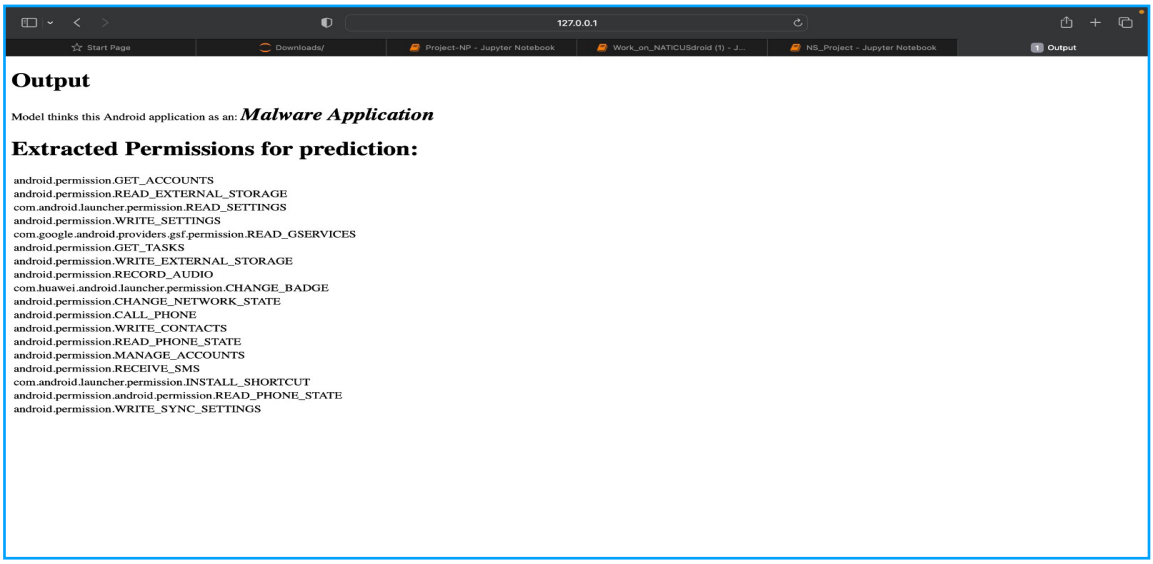


Figure 3: Android Malware Detection

Conclusion:

In this works, Results of Mathur et al., 2021 generated and multi-layer perceptron applied along with LIME to get prediction explanation for selected test instance. To distinguish between harmless apps and harmful software for NATICUSdroid, we also utilized feature selection methods and assessed multiple machine learning (ML) algorithms. Our experiments revealed that the Random Forest, Multilayer Perceptron, Gradient Boosting, and Extreme Gradient Boosting models achieved the highest results. These models recorded performance measures of 97% accuracy, 97% recall, 97% precision, and 97% F1-score, based on our findings.

Contribution of Team Members:

1. Data Gathering and Data Pre-processing: ***Nikhil Panda***
2. Data Processing: ***Sai kiran***
3. Feature Extraction: ***Sai Kiran***
4. Logistic Regression, Extra Tree, Gradient Boosting - ***Nikhil Panda***
5. Extreme Gradient Boosting, KNN, AdaBoost, Support Vector Machine - ***Sai Kiran***
6. Random Forest, Gaussian NB, Multi-Layer Perceptron - ***Likhitha Javvaji***
7. LIME Prediction - ***Likhitha Javvaji, Sai Kiran***
8. Service Development - ***Nikhil Panda, Likhitha Javvaji***
9. Documentation - ***Nikhil Panda, Sai Kiran, Likhitha Javvaji***

References:

[1] https://www.mdpi.com/2227-7390/9/21/2813
[2] https://core.ac.uk/download/pdf/80994982.pdf
[3] https://www.sciencedirect.com/science/article/abs/pii/S0045790617320256
[4] https://gridinsoft.com/blogs/android-malware/
[5] https://dataprot.net/guides/how-to-remove-malware-from-android/
[6] https://www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/
[7] https://scikit-learn.org/stable/supervised_learning.html#supervised-learning
[8] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.RocCurveDisplay.html#
[9] https://medium.com/dataman-in-ai/explain-your-model-with-lime-5a1a5867b423
[10] https://towardsdatascience.com/anova-for-feature-selection-in-machine-learning-d9305e228476
[11] Mathur, A., Podila, L. M., Kulkarni, K., Niyaz, Q., & Javaid, A. Y. (2021). NATICUSdroid : A malware detection framework for Android using native and custom permissions. Journal of Information Security and Applications, 58, 102696.
[12] Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D., & Liu, H. (2020). A review of android malware detection approaches based on machine learning. IEEE Access, 8, 124579-124607.
[13] Kouliaridis, V., & Kambourakis, G. (2021). A comprehensive survey on machine learning techniques for android malware detection. Information, 12(5), 185.
[14] Saad, S., Briguglio, W., & Elmiligi, H. (2019). The curious case of machine learning in malware detection. arXiv preprint arXiv:1905.07573.
[15] El Merabet, H., & Hajraoui, A. (2019). A survey of malware detection techniques based on machine learning. International Journal of Advanced Computer Science and Applications, 10(1).
[15] Seraj, S., Khodambashi, S., Pavlidis, M., & Polatidis, N. (2022). HamDroid: permission- based harmful android anti-malware detection using neural networks. Neural Computing and Applications, 34(18), 15165-15174.
[16] Seraj, S., Pavlidis, M., & Polatidis, N. (2022, June). TrojanDroid: Android Malware Detection for Trojan Discovery Using Convolutional Neural Networks. In Engineering Applications of Neural Networks: 23rd International Conference, EAAAI/EANN 2022, Chersonissos, Crete, Greece, June 17–20, 2022, Proceedings (pp. 203-212). Cham: Springer International Publishing.
[17] Şahin, D. Ö., Kural, O. E., Akleylek, S., & Kılıç, E. (2021). Permission-based Android malware analysis by using dimension reduction with PCA and LDA. Journal of Information Security and Applications, 63, 102995.
[18] Agrawal, P., & Trivedi, B. (2021). Machine learning classifiers for Android malware detection. In Data Management, Analytics, and Innovation: Proceedings of ICDMAI 2020, Volume 1 (pp. 311-322). Springer Singapore.
[19] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).
[20] Das, K., & Behera, R. N. (2017). A survey on machine learning: concept, algorithms, and applications. International Journal of Innovative Research in Computer and Communication Engineering, 5(2), 1301-1309.
[21] Suleman Saleem, M., Mišić, J., & Mišić, V. B. (2022). Android Malware Detection using Feature Ranking of Permissions. arXiv e-prints, arXiv-2201.
[22] ARSLAN, R. S., Ölmez, E., & Orhan, E. R. (2021). Afwdroid: deep feature extraction and weighting for machine. Dicle üniversitesi MÜhendislik Fakültesi Mühendislik Dergisi, 12(2), 237-245.
[23] Malware Tools - https://www.softwaretestinghelp.com/remove-malware-from-android-phone/
[24] NATICUSdroid Dataset - https://archive-beta.ics.uci.edu/dataset/722/naticusdroid+android+permissions+dataset
[25] https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9