Malware Classification of Android Applications Based on Permissions

Vinay Palakurthy, [1] Vikramaditya Gurrapu, [2] Balaji Mediboina, [3] Rachana Tanneeru[4]
Michigan Technological University[1,2,3,4]
palakurt@mtu.edu,[1] vgurrapu@mtu.edu,[2] bmediboi@mtu.edu,[3] tanneeru@mtu.edu[4]

Abstract

In response to the growing threat of malware targeting Android devices, which poses significant risks to user privacy and data security, this project adopts a permissions-based approach to develop a machine learning-based malware detection system. The aim is to contribute to a safer Android environment by providing a robust classification model for detecting malware-infected applications. This project focuses on preprocessing the data and tuning various models like Random Forest, AdaBoost, kNN, and Logistic Regression with feature selection techniques such as Variance threshold and Chi-Square statistic. This project demonstrates that the Random Forest model performs better with an accuracy of 95% for Chi-Square Statistic and 96% for variance threshold than all other models in both feature selection techniques with an accuracy of 95% for Chi-Square Statistic and 96% for variance threshold. The models are evaluated based on accuracy, and further metrics such as precision, recall, f1 score, true positive rate, and false positive rate are calculated to get a better understanding of the models.

Introduction

Android's leading status in the world of smartphones showcases its global appeal and the preference it commands among users [1]. As of March 2024, Android boasts a remarkable market share of 70.79% in the mobile platform, according to StatCounter GlobalStats [2]. This popularity is also because of its open-source nature and easy customizability [3]. However, this status has also made Android devices prime targets for malware attacks. Malicious software, or malware, comprises various harmful programs that access or damage devices without authorization, threatening users' privacy, data integrity, and the functionality of their devices. Android's flexibility, including its extensive app ecosystem and the ability to install apps from unofficial sources, further amplifies these security vulnerabilities.

Given that the average smartphone user interacts with 10 apps daily and 30 apps monthly [4], the potential for malware exposure escalates, emphasizing the urgency for robust security measures. The evolving sophistication of malware, which often manipulates app permissions to infiltrate user data and system resources, highlights the challenge of developing efficient detection tools. Such security threats not only compromise individual user safety but also pose risks to the broader digital ecosystem, as infected devices can become part of botnets for further malicious operations.

Given these concerns, our project introduces a permissions-based malware detection classification model that leverages machine learning using the NATICUSdroid Android Permissions Dataset [5]. This dataset consists of permissions from more than 29000 benign & malware Android apps released between 2010-2019 and serves as the foundation for our analysis. We checked for any missing values and near-zero variance features. Then, we performed feature selection techniques, namely, the Chi-Square Statistic and Variance threshold, to reduce the number of input variables by removing irrelevant features. Once this step was completed, we used the resampled and 10-fold cross-validated data to examine the predictive performance of various classification models with optimized hyperparameters. These included linear and non-linear models such as logistic regression, kNN, AdaBoost, and Random Forest [5][6]. Evaluation metrics like accuracy, precision, recall, and F1-score were considered to define the best model for predicting whether an app is benign or malware.

Overall, in addressing the challenge of predicting whether an app is benign or malware, Random Forest was identified as the best model as it had better accuracy than all the other models evaluated. We also examined the top 15 permissions the Random Forest model uses to determine if an app is benign or malicious.

Related Work

The field of Android malware detection has gained much attention in recent years due to the growing number of mobile devices. Several strategies have been deployed to classify Android apps as malicious or benign, notably involving machine learning algorithms. Commonly used techniques include decision trees, support vector machines, and logistic

regression, which analyze features to make classifications [7]. Using a new method, PUMA: Permission Usage to Detect Malware in Android Sanz et al. [8] achieved an accuracy of 86.41%. Xu et al. [9] identified malware applications using graph-based representations to attain an accuracy of 85.3%. In Addition, Vinayakumar et al. [10] classified Android permissions using deep learning methods to achieve 89.7% accuracy. Furthermore, Turnip et al. [11] using extreme gradient boosting attained 75.5% accuracy. All these techniques do not cross over 90% of accuracy due to lack of a few preprocessing techniques like feature selection.

All the above-mentioned techniques do not attain high accuracy. So, we performed two different feature selection techniques to remove irrelevant features and just keep the required features. This addresses dimensionality issues as well as making the processes more accurate. Also, we performed hyperparameter tuning for every model to get optimized parameters.

Data

We're utilizing the NATICUSdroid Android Permissions Dataset from the University of California, Irvine Machine Learning Repository, designed to facilitate the development of machine learning models aimed at distinguishing between benign and malicious Android applications [5]. The dataset includes permissions from over 29,000 Android apps released between 2010 and 2019. This dataset contains information about whether an app is benign or malware, making it suitable for our classification models. It has 86 features, representing the characteristics or attributes of the apps.

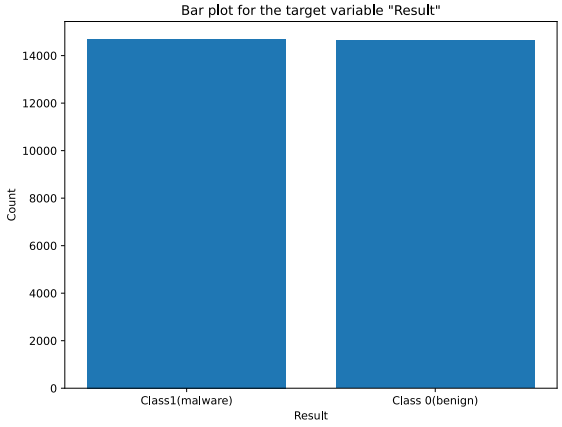

Figure 1: Distribution of Target variable

The structure of the permissions dataset is binary and nominal. Each permission is represented by a binary value: 1 or 0, where 1 indicates that specific permission has been requested by an app, and 0 denotes its absence.

Initially, we checked for any incomplete or inconsistent entries and near-zero variance and found that the dataset neither had missing values nor near-zero variance values. Since the response variable is categorical, assessing the balance of the dataset was crucial. We found the response variable in our dataset is balanced (the percentage of data with class 0 is 49.88% and the percentage of data with class 1 is 50.12%), as illustrated in Figure 1.

Methods

In this section, we discuss the feature selection techniques and classification methods used for the classification of apps into benign or malware.

Feature Selection Techniques

Feature selection is an important step in identifying the most relevant features from the dataset to improve model performance and reduce complexity. We have used two feature selection techniques, namely variance thresholding and Chi-Square Statistic.

Variance Thresholding

This technique involves removing features whose variance does not meet a certain threshold. It assumes that features with low variance are less informative or relevant for classification tasks. In our context, this approach can help identify permissions that are very common or rare across apps, thus unlikely to contribute significantly to distinguishing between benign and malicious applications. We specified that features with a variance less than 0.05 would be removed. Equation for variance threshold:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Chi-Square Statistic

The Chi-Square test is used to determine the independence of two variables making it ideal for categorical data. By applying this test, we identify and retain features that have a significant relationship with the target variable (benign vs. malicious). The principle behind the Chi-Square test is to compare the differences between the observed frequencies of feature occurrences across classes and the expected frequencies if the feature and target were independent of each other. A high Chi-Square value indicates that the observed and expected frequencies differ significantly, suggesting a strong association between the feature and the target class. This process selects the top k features that have the highest Chi-Square values concerning the target variable. In our case, k is set to 25, meaning that the 25 features most strongly associated with the app being benign or malicious are selected for further model building.

Pearson's Chi-Square test formula:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Classification Models

For, our project, we examined both linear and non-linear classification models. We chose the logistic regression model for the Linear model, and for the non-linear models, we selected kNN, AdaBoost, and Random Forest.

Logistic Regression

Logistic regression is a supervised machine-learning technique commonly used for binary classification applications. It is also termed a discriminative model, which implies it aims to differentiate between classes (or categories). This method uses the logistic (or sigmoid) function to convert a linear combination of input data into a probability value between 0 and 1. This probability represents the possibility that a given input falls into one of two defined categories.

Logistic regression is used to assess the connection between a dependent variable and one or more independent variables; however, it is used to predict a categorical variable rather than a continuous one.

K-Nearest Neighbors (kNN)

K-nearest Neighbors (KNN) is a non-parametric supervised machine learning approach used to solve classification as well as regression issues.

kNN is a classification technique in which the function is only approximated locally and all computation is postponed until function evaluation. Because this method uses distance for classification, if the features represent distinct physical units or arrive at highly different sizes, normalizing the training data can significantly increase its accuracy.[12]

AdaBoost

AdaBoost, or Adaptive Boosting, is a statistical classification meta-algorithm. It may be used in combination with a variety of different learning algorithms to boost performance. The output of the other learning algorithms ('weak learners') is merged to form a weighted total, which reflects the boosted classifier's final output. AdaBoost is often used for binary classification, although it may be extended to multiple classes or limited intervals on the real line.[13][14]

AB is particularly sensitive to noise and outliers in data sets. It is less vulnerable to overfitting than other types of learning algorithms in certain situations. AB is largely considered the superior out-of-the-box predictor.

Random Forest

Random forest is a popular machine-learning technique developed by Leo Breiman and Adele Cutler that integrates the outputs of numerous decision trees to produce a single outcome.

The random forest technique is an extension of the bagging method since it uses both bagging and feature randomization to generate an uncorrelated forest of decision trees. Feature randomization, also known as feature bagging or "the random subspace method", creates a random selection of features to ensure minimal correlation among decision trees.

For regression tasks, the mean or average forecast of each tree is returned. Random decision forests address decision trees' tendency to overfit their training set.[14]

Experiments and Results

Experimental Design

In Feature selection, a 0.05 threshold value was used for variance thresholding, which removes features with variance less than 0.05 and 25 was selected as k value for the Chi-square statistic, which selects the top 25 features strongly associated with the target variable. After feature selection techniques, the dataset was initially divided into two parts: a training/ validation set and a test set. This split was performed with a test size of 20% of the data, ensuring stratification to maintain the same proportion of benign and malicious apps. The training/ validation set was then further divided into a training set and a validation set, this time with a validation size of 25% of data, with stratification to ensure a consistent distribution of classes.

The training set was used to train the models, the validation set was used to tune the hyperparameters and make decisions about the models without bias, and the test set provided an unbiased evaluation of the final model performance. Grid search was used to tune hyperparameters such as regularization parameter 'C' and the type of regularization in logistic regression, to explore different values of 'k' in kNN, tune weak classifiers in AdaBoost, and adjust the number of trees, maximum depth of trees for the random forest.

Using the best parameters obtained from grid search, classification models including logistic regression, kNN, AdaBoost, and random forest were trained. These models have been done on different feature sets obtained from feature-selection algorithms like variance thresholding, and the Chi-Square test.

Metrics

Given the balanced nature of our dataset, we chose accuracy as the evaluation metric. In the Logistic Regression analysis, models employing L2 regularization and Elastic Net regularization with a hyperparameter C value of 1 showcased the highest accuracy levels, at 0.94256 and 0.9463 respectively. The former utilized the Chi-Square Statistic for feature se-

lection, while the latter applied Variance Thresholding. Figure 2 and Figure 3 demonstrate the consistency of the Logistic regression model accuracy across various C values, indicating the robustness of our model against hyperparameter fluctuations. This highlights the effectiveness of our regularization strategies in enhancing predictive performance.

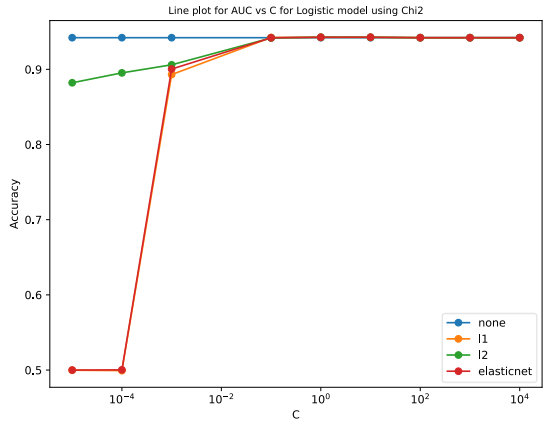

Figure 2: Hyperparameter tuning plot of Logistic regression model – Chi-Square Statistic

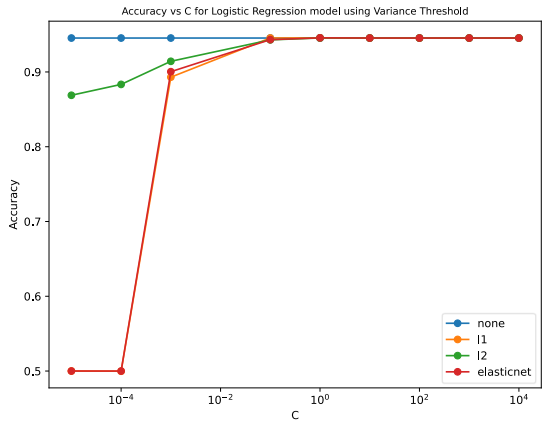

Figure 3: Hyperparameter tuning plot of Logistic regression model – Variance threshold

For the k-nearest Neighbors (kNN) model, the plot illustrates the impact of the number of neighbors on model accuracy, utilizing two feature selection techniques: Chi-Square Statistic and Variance Thresholding. The optimal number of neighbors identified for the Chi-Square Statistic method shown in Figure 4 is 11, with an accuracy of 0.9493, while for the Variance Threshold method shown in Figure 5, 6 neighbors yield the best accuracy of 0.9590. These plots underscore the importance of feature selection and the choice of k in the performance of the kNN model for classifying Android applications.

The AdaBoost classifier revealed notable accuracy. With 15 estimators, the Chi-Square Statistic approach shown in Figure 6, yielded an accuracy of 0.9396. On the other hand, the Variance Threshold approach shown in Figure 7, peaked at an accuracy of 0.9423 with 18 estimators. These results, represented in the line graphs, demonstrate how varying the number of estimators influences the model's predictive power, with a discernible trend towards higher accuracy with an increased number of estimators up to a point of stabilization.

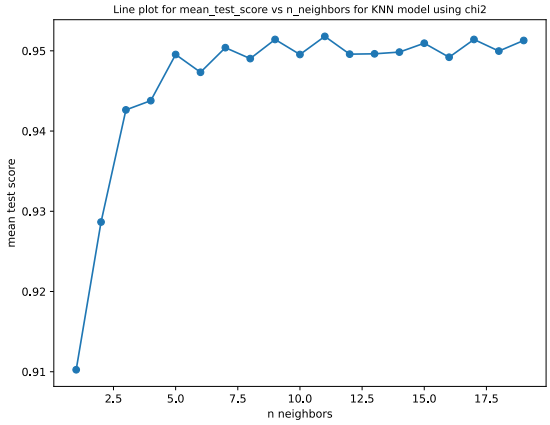

Figure 4: Hyperparameter tuning plot of kNN model – Chi-Square Statistic

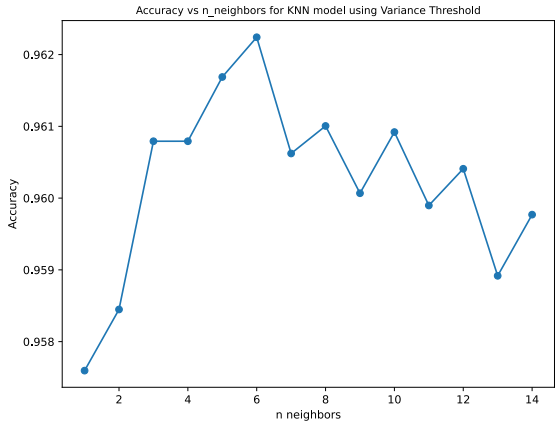

Figure 5: Hyperparameter tuning plot of kNN model – Variance threshold

With Chi-Square Statistic, the random forest model shown in Figure 8 reached an accuracy of 0.9544 using 28 estimators and a maximum of 5 features. On the other hand, with Variance Threshold shown in Figure 9, the accuracy peaked at 0.9618 with 47 estimators, also considering a maximum of 5 features. The line graphs show the performance of the Random Forest model as the number of estimators changes, showing a clear trend where accuracy improves with more estimators to a point of optimal balance.

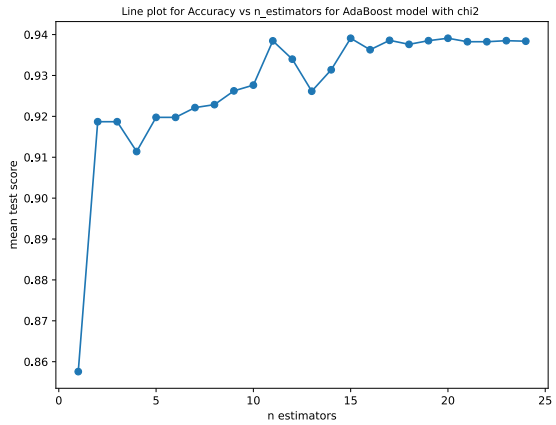

Figure 6: Hyperparameter tuning plot of AdaBoost model – Chi-Square Statistic

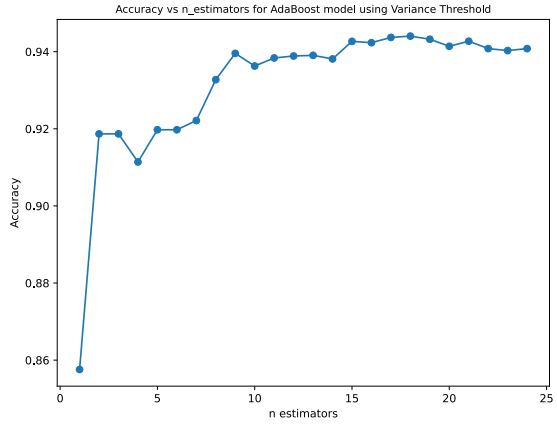

Figure 7: Hyperparameter tuning plot of AdaBoost model – Variance threshold

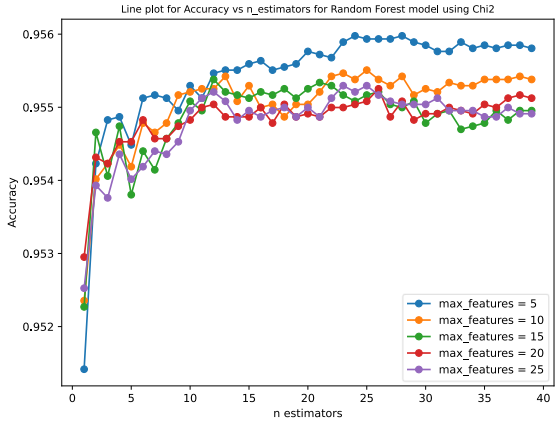

Figure 8: Hyperparameter tuning plot of Random Forest model – Chi-Square Statistic

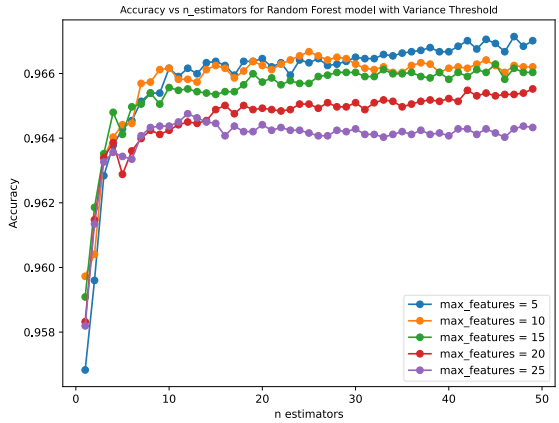

Figure 9: Hyperparameter tuning plot of Random Forest model – Variance threshold

Among all tested models, the Random Forest classifier was the best model, which is shown in Table 1.

Classification model	Chi-square statistic	Variance Threshold
Logistic Regression	0.9425	0.9463
kNN	0.9493	0.9590
AdaBoost	0.9396	0.9423
Random Forest	0.9544	0.9618

Table 1: Comparison of Model Accuracy Scores Using Chi-Square Statistic and Variance Threshold Feature Selection Methods

Results

We further evaluated the performance of the best model using additional metrics, including the confusion matrix, F1 score, precision, and recall.

The confusion matrices for the Random Forest model with both Chi-Square statistic Figure 10 and Variance Threshold Figure 11 feature selection highlight its classification accuracy. The high number of true positives and true negatives indicates a strong ability to correctly identify both classes. The low number of false positives and false negatives underscore the model's precision and recall in distinguishing between benign and malicious apps, supporting the model's high F1 scores as reflected in Table 2.

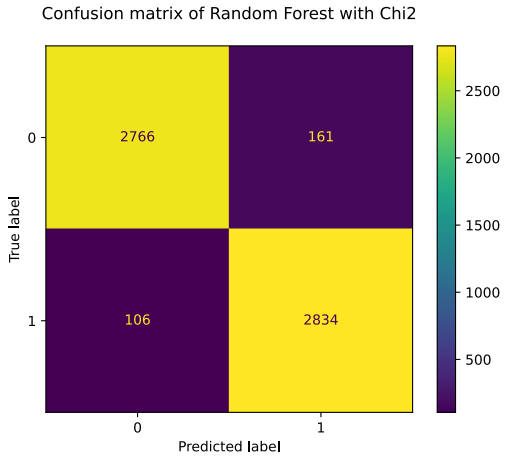

Figure 10: Confusion Matrix of Random Forest model – Chi-Square Statistic

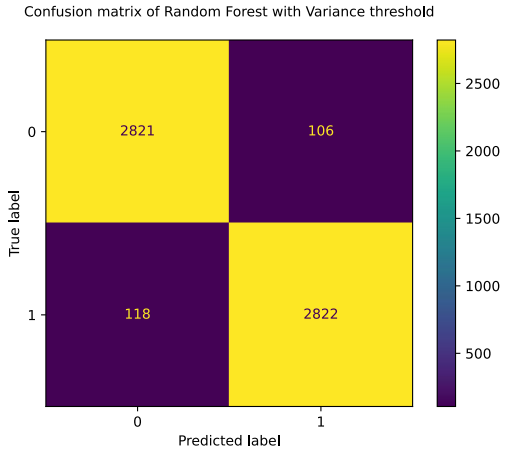

Figure 11: Confusion Matrix of Random Forest model – Variance threshold

Best model metrics	Chi-square statistic	Variance Threshold
F1 Score	0.9550	0.9618
Precision	0.9462	0.9638
Recall	0.9639	0.9599
TPR	0.9639	0.9599
FPR	0.0550	0.0362

Table 2: Random Forest Model Performance Metrics Using Variance Threshold and Chi-Square Feature Selection Methods

The True Positive Rate (TPR) echoes the high recall, demonstrating the model's effectiveness in identifying actual positives. The False Positive Rate (FPR) is relatively low, further underscoring the model's accuracy in distinguishing between classes. The metrics across both feature selection methods suggest that the model has a robust predictive capability.

The bar charts delineate the top 15 features that the Random Forest model deemed most significant in distinguishing between benign and malicious apps using the Chi-Square Statistic Figure 12 and Variance Threshold methods Figure 13. Permissions like Read phone state and Write external storage rank high in importance across both feature selection techniques. This concurs with the evaluation metrics showcased in Table 2, where both models exhibit high accuracy, precision, recall, and F1 scores, indicating a strong predictive performance based on these key permissions.

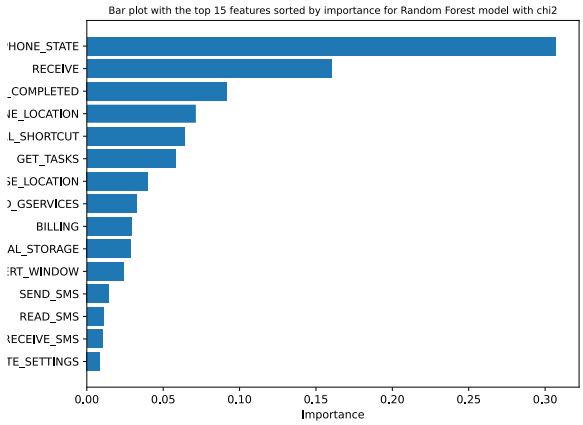

Figure 12: Important Features of Random Forest model – Chi-Square Statistic

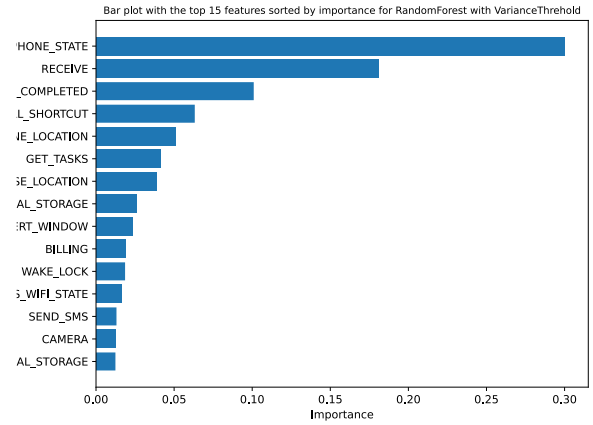

Figure 13: Important Features of Random Forest model – Variance threshold

Conclusion

In this study, we aimed to enhance Android malware detection using permissions data from over 29,000 apps, employing machine learning techniques like logistic regression, k-Nearest Neighbors, AdaBoost, and Random Forest. Through preprocessing, feature selection, and hyperparameter optimization, we aim to identify permissions critical for distinguishing between benign and malicious apps and identified Random Forest performs better than all other models with an accuracy of 0.9544 with Chi-square statistic feature selection. On the other hand, with Variance Thresholding our Random Forest model performed slightly better with an accuracy of 0.9618. One limitation of this work is the reliance on permissions data alone, which while informative might not capture all behavioral nuances of malicious applications. The future scope of this project could involve integrating additional types of data, such as API calls or runtime behavior. This research contributes to cybersecurity efforts by demonstrating the viability of machine learning in identifying malicious applications, with the potential for further improvement and real-world application.

References

1] "Usage share of operating systems." Wikipedia, The Free Encyclopedia. Wikimedia Foundation, https://en.wikipedia.org/wiki/Usage_share_of_operating_systems. Accessed: 2024-04-20.

2] Statcounter - GlobalStats, Mobile operating system market share worldwide, Statcounter - GlobalStats, 2019. https://gs.statcounter.com/os-market-share/mobile/worldwide. Accessed on: 2024-03-28.

3] Android (Operating System). Wikipedia, The Free Encyclopedia, Wikimedia Foundation, Inc., https://en.wikipedia.org/wiki/Android_(operating_system). Accessed: 2024-04-20."

4] How Many Apps Do People Really Use on a Regular Basis?" LinkedIn, https://www.linkedin.com/pulse/how-many-apps-do-people-really-use-regular-basis-codebase-one. Accessed: 2024-04-20.

5] A. Mathur, L.M. Podila, K. Kulkarni, Q. Niyaz, A.Y. Javaid, NATICUSdroid: A malware detection framework for Android using native and custom permissions, Journal of Information Security and Applications, 58 (2021) 102696.

6] H. Bai, N. Xie, X. Di, Q. Ye, FAMD: A Fast Multifeature Android Malware Detection Framework, Design, and Implementation, IEEE Access, 8, 2020, pp. 194729-194740, doi: 10.1109/ACCESS.2020.3033026.

7] Zaki, Mohammed J., and Wagner Meira. Data mining and analysis: fundamental concepts and algorithms. Cambridge University Press, 2014.

8] Sanz, Borja, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, Pablo Garcia Bringas, and Gonzalo Álvarez. "Puma: Permission usage to detect malware in android." In International joint conference CISIS'12-ICEUTE 12-SOCO 12 special sessions, pp. 289-298. Springer Berlin Heidelberg, 2013.

9] Xu, Lifan, Dongping Zhang, Marco A. Alvarez, Jose Andre Morales, Xudong Ma, and John Cavazos. "Dynamic android malware classification using graph-based representations." In 2016 IEEE 3rd international conference on cyber security and cloud computing (CSCloud), pp. 220-231. IEEE, 2016.

10] Vinayakumar, R., K. P. Soman, and Prabaharan Poornachandran. "Deep android malware detection and classification." In 2017 International conference on advances in computing, communications, and informatics (ICACCI), pp. 1677-1683. IEEE, 2017.

11] Turnip, Togu Novriansyah, Amsal Situmorang, Ayu Lumbantobing, Josua Marpaung, and Samuel IG Situmeang. "Android malware classification based on permission categories using extreme gradient boosting." In Proceedings of the 5th International Conference on Sustainable Information Engineering and Technology, pp. 190-194. 2020.

12] Hastie, Trevor, Robert Tibshirani, and Jerome H. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York: Springer, 2001.

13] Freund, Yoav, and Robert E. Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting." In Lecture Notes in Computer Science, edited by Springer Berlin Heidelberg, 23–37. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. doi:10.1007/3-540-59119-2_166.

14] Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. New York: Springer, 2009. ISBN 978-0-387-84858-7.