Android Authenticity Prediction

Vinayak Marathe, Riya Patel

Abstract - Mobile phones are increasingly a target of attack by malicious software. Android application vendors have developed several techniques to protect users from such malicious applications, including the use of an authentication scheme. Authentication schemes are used to verify the integrity of applications before they are installed or launched on a device. In this paper, we propose an Android Authenticity Prediction (AAP) system to predict the authenticity of applications before they are allowed to run on the user's device. The AAP system is based on machine learning and uses application features extracted from static analysis of the application code. We validate the AAP system on a dataset consisting of benign (0) and malicious (1) applications. The system is based on machine learning Classification Algorithms and uses application features extracted from static analysis of the application code.

Keywords: Prediction algorithms, Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, XGBoost, Naïve Bayes, KNN, SVM, Android, Malware, Benign, Malicious

I. INTRODUCTION

The first Android smartphone was launched in September 2008, and shortly thereafter, smartphones powered by the new open-source operating system were everywhere. In 2021, almost 12 new enhanced versions of Android were released, and it is the most widely used mobile operating system in the world, with an 84% share of the global smartphone market. With this level of adoption coupled with the open-source nature of Android applications, security attacks are becoming more and more ubiquitous and seriously threaten the integrity of Android applications. Statistics show that more than 50 million malware and potentially unwanted applications have been identified for Android.

The proposed Android Authenticity Prediction (AAP) system uses machine learning techniques to classify applications as authentic or malicious based on a set of application features extracted from static analysis of the application code. Once the model is trained, it can be used to predict the authenticity of the mobile applications. Exploratory data analysis and feature engineering are first implemented on each dataset to reduce a large number of features available and we are focusing on our variables and understanding the behavior of our variables. Then, on the next stage, Hypothesis Testing is performed. We defined three hypothetical statements from the dataset. And in those three questions, we obtain final conclusions about the statements through the code and statistical testing. And finally we have used different supervised classification models to classify whether a given app is malware or benign. The performance metrics of different models are then compared to identify the technique that offers the best results for this purpose of malware detection. In the end and final stage of the project, we have implemented our model using different types of classification algorithms which are further evaluated using different evaluation techniques in order to check the accuracy of our model and then get the report back with the maximum accurate model. In the end, we have come up with the achieved business insights and conclusion of our project.

II. Problem Statement

The problem of Android Authenticity Prediction is to develop a machine learning model that can accurately predict whether an Android application (app) is authentic or not. With the increase in the number of mobile apps, the risk of downloading malicious apps has also increased. Malicious apps can steal sensitive user data, perform unwanted actions, and damage the user's device. Therefore, it is essential to develop a model that can accurately predict the authenticity of Android apps and

help users make informed decisions about which apps to download and install on their devices. The challenge is to identify the relevant features that can distinguish between authentic and malicious apps and to train a classification model that can generalize well to new, unseen apps. Additionally, the model should be able to handle the large and dynamic nature of the mobile app ecosystem, where new apps are constantly being developed and released.

III. METHODOLOGY

The proposed methodology's implementation begins with downloading the dataset. Then data wrangling and feature manipulation is executed as a step of pre-processing of data. After this, the data is analyzed and a different model is executed. Then we have done the hypothesis testing with 3 different hypothetical statements. At last, all the business insights carried out in this project.

A. Datasets

By using publicly available labeled data sources, different classification models are built to distinguish between malware and benign applications. The dataset consists of every expertise that is helpful and attributes that are no longer useful. So, in pre-processing the beneficial statistics is chosen and statistics cleansing is performed to get the best possible attributes. Our dataset has 183 independent features and 1 dependent feature which is also a target variable. Our data has a huge number of features with a shape of (29999, 184) i.e. 29,999 rows and 184 columns. Let's understands some of the important features from the bundle of 184 features:

- **App** - Name of the App
- **Package** - OBB/Data package installed in root folder
- **Category** - App Category (e.g. Entertainment, Adventure, puzzle, Action, Antivirus, etc.)
- **Description** - App Description
- **Rating** - Rating out of 5
- **Number of ratings** - No. of Ratings given by users
- **Price** - Price of the App
- **Related apps** - Apps related to installed App
- **Dangerous (D) permissions count** - No. of Dangerous Permissions allowed by user
- **Safe (S) permissions count** - No. of Safe Permissions allowed by user where the meter was disengaged
- **Class(Type of app)** - 0: Benign, 1: Malware

B. Data Wrangling & Visualization

The quality of data performs a fundamental role, and the most cautiously depicted trouble to be. For this research, as a part of data pre-processing, we have identified the irrelevant or redundant data: such as duplicate observations, entries with missing or invalid values, or data that is outside of the scope of the project and worked on it pretty well by ensuring that all data is in the correct format. Next, we have visualized our relationship between independent and dependent features graphically (as seen in the fig.1 below) as the dataset is in tabular shape and it is hard to appear at and understand the data in this or any other way. Data visualization helps in grasping the style of the data. Data visualization in this method is a graphical illustration of the data. In this analysis, the utilization of bar charts, dist. plots, pie charts, histograms and sub plots, the cleaned records obtained thru pre-processing is visualized. From one of the visualizations, we got to know about the dataset that the dependent or target variable is imbalanced so we have to handle class imbalances further. Visualization makes it easy to maintain the attribute's tough relationship with the beneficial resource of graphical representation.

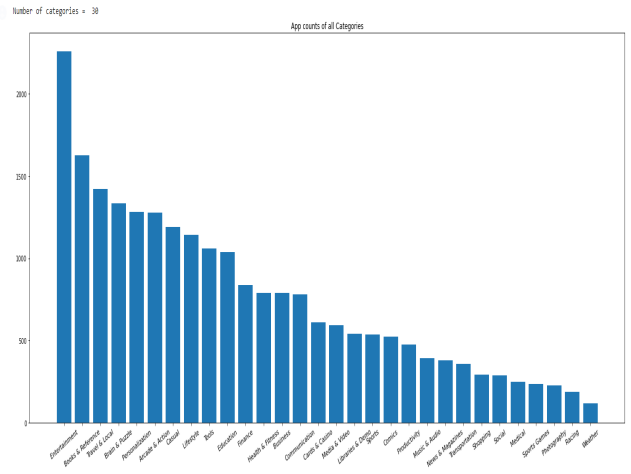

Fig.1 EDA based on attributes

C. Hypothesis Testing

We have defined three hypothetical statements from the dataset. In the next three questions, perform hypothesis testing to obtain a final conclusion about the statements through our code and statistical testing.

1. The app's rating is a significant predictor of its authenticity. Here,
    - **Null hypothesis:** The app's rating is not significantly related to its authenticity.
    - **Alternative hypothesis:** The app's rating is significantly related to its authenticity.
    - **Result:** The output suggests that the app's rating is a significant predictor of its authenticity, and the mean rating of authentic apps is significantly higher than that of fake apps. That's why we have to reject the null hypothesis.

2. The app's category is a significant predictor of its authenticity. Here,
    - **Null hypothesis:** The app's category is not significantly related to its authenticity.
    - **Alternative hypothesis:** The app's category is significantly related to its authenticity.
    - **Result:** The output suggests that the app's category is a significant predictor of its authenticity, and there is a significant relationship between app category and authenticity and that's why we have to reject the null hypothesis.

3. The number of reviews is a significant predictor of an app's authenticity. Here,
    - **Null hypothesis:** The number of reviews is not significantly related to an app's authenticity.
    - **Alternative hypothesis:** The number of reviews is significantly related to an app's authenticity
    - **Result:** The result of the chi-square test in the code provided indicates that the p-value is 0.0, which is less than the typical significance level of 0.05. This means that we reject the null hypothesis that the number of ratings is not significantly related to the Class of the app and conclude that there is a significant relationship between the number of ratings and the Class of the app.

D. Feature Manipulation & Selection

In feature engineering, we have done some manipulation with our dataset. Following is the steps we have performed on our dataset to get the proper insights:

- **Handling Missing Values –** Generally, there are two approaches to handle the missing values i.e. deleting the Missing values and imputing the Missing Values. We have used both of the methods here. For the Apps column where one row has null values, we used the first approach that is deleting the missing values. While another column has more rows with null values so there we have used a second approach of imputing the null values. In the Description column, we are imputing the null values with arbitrary values and making an educated guess about the missing value with 'Not Available' tag. Next, in the related apps column, we have filled the values with a string commenting 'No related app found'. Last and important column, Dangerous Permission Count has a maximum number of missing values with 755 rows so here we have used the second approach where we have imputed the missing value with the median. The median is the middlemost value. It's better to use the median value for imputation in the case of outliers.

- **Categorical Encoding –** As our dataset is very huge with a shape of (29999, 184), it also has some categorical column named 'Category' which is also

an important column for our analysis so we have encoded this columns using a technique called one hot encoding is used to represent categorical variables as numerical values in a machine learning model. And after handling missing values and categorical columns, we found the shape of our dataset to be (22823, 213).

- **Handling Data Imbalanced –** As our project is a classification problem, we have to check for a Handling Data Imbalanced that is checking labels to handle the highly imbalanced data, if any. Our target variable is highly imbalanced here so we have used some techniques which are as followed –
    - **Resampling Techniques -** A widely adopted technique for dealing with highly unbalanced datasets is called resampling. It consists of removing samples from the majority class (random under-sampling) and/or adding more examples from the minority class (random over-sampling).
    - **Tomek Links -** Tomek links are pairs of very close instances but of opposite classes. Removing the instances of the majority class of each pair increases the space between the two classes, facilitating the classification process. Tomek's link exists if the two samples are the nearest neighbors of each other.
    - **Synthetic Minority Oversampling technique (SMOTE) -** This technique generates synthetic data for the minority class. SMOTE works by randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors.
    - **Penalize Algorithms (Cost Sensitive Training) -** The next tactic is to use penalized learning algorithms that increase the cost of classification mistakes on the minority class. A popular algorithm for this technique is Penalized-SVM. During training we can use the argument class_weight='balanced' to penalize mistakes on the minority class by an amount proportional to how under-represented it is.

- **Feature Manipulation and Selection –** In this section, we have manipulated features and dropped some unnecessary columns which are of literally no use is then selected and performed mutual information. It calculates mutual information value for each of independent variables with respect to dependent variables, and selects the ones which have the most information gain (we will see it further in IV section of model explainability). The top 10 features in our data which will be used in our classification ML models are - Number of ratings, Price, Rating, Category_Travel & Local, Category_Comics, Network communication: view network state (S), Category_Libraries & Demo, Safe permission count, Category Cards & Casino and Category_Health & Fitness. After this, we have splitted further into train and test data for the purpose of implementation.

IV. MODEL IMPLEMENTATION

Considering various Machine Learning models that provide reliable and improved accuracy for prediction based use-cases, Logistic Regression, Decision Tree, Random Forest, Gradient Boost, XGBoost, Naïve Bayes, KNN and SVM are taken into consideration.

A. **Logistic Regression** – Logistic regression is a type of machine learning algorithm that is used in supervised learning to predict the likelihood of an event occurring. It is based on the assumption that there is some sort of relationship between the input features and the output variable, i.e. the probability of a certain event occurring.

By training the algorithm on data related to particular characteristics of Android devices, it is possible to predict whether a specific device is genuine or not with a 65.02 accuracy before Cross validation and hyper-parameter tuning which is good. After applying Cross validation and hyper-parameter tuning, we have found the test accuracy of **70%**.

B. **Decision Tree** – Decision tree algorithm is a supervised learning approach used for classification and regression tasks. It works by creating a tree-like structure of decisions, where each node in the tree is a decision point. At each decision point, the best feature from the data set is chosen and split into two branches based on the outcome of the split. This process is repeated until the leaf node is reached, at which point the final outcome is determined. The accuracy of the model is evaluated on a test data set to ensure that the model is able to predict the correct outcome with a high degree of accuracy. We have used DT with criterion entropy and a depth of 25 subtree and found a quite good accuracy of **77.32%** with the test data and AUCROC Score of **63.50%**.

C. **Random Forest** – Random Forest is a supervised machine learning algorithm used for both regression and classification tasks. It is an ensemble learning technique which uses multiple decision trees to obtain better predictive performance. Random Forest algorithms create decision trees on randomly selected data samples, get predictions from each tree, then average the results to reduce the variance for improved accuracy. This technique is particularly useful for dealing with large datasets and high-dimensional input spaces. The main benefits of using Random Forest algorithms include that they are simple to implement, easy to use, and require less tuning than other complex algorithms. Random Forest also offers high accuracy and can protect against overfitting by randomizing samples, reducing variance, and increasing generalization of the model. We have found our model accuracy carried out to be an Accuracy Score of 84.39%. After performing the cross validation and hyper parameter tuning, we have found accuracy to be **85.11%** with precision and recall of 0.90 & 0.87 and most important Auc_Roc_Score of **63.50%**. But when we used random forest with tomek links technique, we found the auc roc score of **86.23%** and with random forest using SMOTE, we got the auc roc score of **88.24** which is the best among all the models.

D. **Naïve Bayes** - Naive Bayes is a family of algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. It is primarily used for classification tasks and is often successful in particular with a small amount of data. It assumes that the occurrence of a particular feature is independent of the occurrence of other features, which simplifies the calculation of the probability of an event. With the help of this algorithm, we found that our model has accuracy of 64.85% before cross validation and tuning which is absolutely not good. Final accuracy we have got is the same as before CV **66.85%.**

E. **K-Nearest Neighbor** – K Nearest Neighbor (KNN) is a supervised machine learning algorithm used for classification and regression. It is a non-parametric and lazy learning algorithm. The KNN algorithm uses a training set of data to classify new data points based on the similarity of its neighbors. It stores all available cases and classifies new cases by a majority vote of its neighbors. The

odds of the closest neighbor is given higher weights than those further away. This model gave us an accuracy score of 75.79% with precision and recall of 0.83 and 0.72 before cross validation and hyper parameter tuning. In the end, we got the accuracy of **77.27%** with auc roc score of **77.26%** which is good and considerable.

F. **Gradient Boosting –** Gradient boosting is an approach to machine learning that combines the output of multiple weak learners in order to create a powerful predictive model. The algorithm works by fitting a new regression or classification model to the residual errors of the previous regression or classification model. This process is iterative, with each successive model improving on the last until a certain criteria is met. It is an efficient algorithm which results in a highly accurate predictive model. Gradient boosting can be used for both supervised and unsupervised learning problems, such as regression and classification. Here, we are using GBC for a classification problem and this bagging method is giving us quite good results of 87.34% is very good but when we tried cross validation and hyper-parameter tuning we got the accurate result of **88.34%** and area under the curve with **95%.**

G. **XGBoost –** XGBoost, which stands for extreme gradient boosting, is an algorithm that uses gradient boosting techniques to create highly accurate predictive models. It is a powerful machine learning algorithm often used in supervised learning problems with both classification and regression problems. It focuses on optimizing the performance of a single model and can be used for both regression and classification tasks. It is well known for its high accuracy and speed. The features that make XGBoost perform well are its parallelization capabilities, built-in regularization and optimization of tree construction, as well as its robustness against over-fitting. XGBoost can also be used for feature selection to reduce the dimensionality. But here we have used xgboost algorithm for the accuracy of the model so we have applied different hyper parameters and performed a cross validation where we found the accuracy of **88.26%** which is quite good.
But when we used XGBoost with tomek links technique, we found the auc roc score of **86.23%** and with XGBoost using SMOTE, we got the auc roc score of **89.92%** which is the best among all the models.

H. **Support Vector Machine –** In this classification algorithm, the SVM algorithm aims to find a decision boundary that separates the data points into two or more classes by maximizing the margin between them. The algorithm attempts to build a decision boundary such that it is maximally distant from all data points, and this is known as the maximum margin classifier. To build a precise decision boundary, SVMs use kernel functions which map the data points into a higher dimensional space, making it easier to identify a decision boundary which does not need to be linear in the original space. SVMs can also handle non-linear decision boundaries, which is why they are so popular. When SVC is used with SMOTE technique, we got the accuracy of **52.17%** auc roc score and **69.04%** accuracy score.

V. MODEL EXPLAINABILITY

In the Android Authenticity Prediction classification project, the model which we have used and found the feature importance using shap model explainability tool to explain predictions.

The feature importance analysis has identified that the **number of ratings** is the most important feature followed by rating and price. This indicates that these features play a significant role in predicting the authenticity of an Android app.

The number of ratings is the most important feature because it is an indicator of the popularity and usage of the app. A higher number of ratings suggest that more people have used the app, which could imply that the app is benign and trustworthy. Additionally, a higher number of ratings also implies that the app has been in the market for a while and has been downloaded by a significant number of users, which could indicate that it is not a new app developed solely for malicious purposes.

The second most important feature is rating, which represents the overall user rating of the app. A higher rating indicates that the app has received positive feedback from users and is likely to be trustworthy. On the other hand, a lower rating could imply that users have had negative experiences with the app and may have reported it as untrustworthy or malicious.

Finally, the third most important feature is price, which represents the cost of the app. This feature may be important because it can serve as an indicator of the app's legitimacy. A very low or very high price compared to other apps in the same category may suggest that the app is not benign or may have been developed with malicious intent.

Overall, these three features provide valuable information for predicting the authenticity of an Android app, and their importance highlights the importance of user ratings, popularity, and pricing in assessing the trustworthiness of an app.

VI. RESULT

The result of all the techniques used for Android authenticity prediction would depend on the data that is used to train the model. If the model has access to accurate and comprehensive data about Android apps, it can be very effective in accurately and precisely predicting whether an Android application is authentic or not. As a result, we found that random forest classifier and xgboost classifier has the highest accuracy among all the techniques we have used so far. XGBoost wins the case here by giving the high number of accuracy that is **89%**.
Follow is the table for Precision, Recall and F1-score of all the important techniques:-

	Random Forest (Using SMOTE)	**XGBoost Classifier (Using SMOTE)**	Gradient Boost
Precision	**0 - 0.86** **1 - 0.90**	**0 - 0.87** **1 - 0.93**	0 - 0.86 1 - 0.91
Recall	**0 - 0.91** **1 - 0.86**	**0 - 0.93** **1 - 0.87**	0 - 0.92 1 - 0.85
F1-Score	**0 - 0.89** **1 - 0.88**	**0 - 0.90** **1 - 0.90**	0 - 0.89 1 - 0.88
	Logistic Regression	Decision Tree	K-Nearest Neighbor
Precision	0 - 0.56 1 - 0.75	0 - 0.64 1 - 0.86	0 - 0.71 1 - 0.89
Recall	0 - 0.44 1 - 0.83	0 - 0.73 1 - 0.80	0 - 0.92 1 - 0.63
F1-Score	0 - 0.49 1 - 0.79	0 - 0.68 1 - 0.83	0 - 0.80 1 - 0.73

Table – 1 Summary of Results

VII. FUTURE WORK

Future directions for improving the android authenticity prediction:

1. **Using advanced machine learning techniques:** Advanced machine learning techniques, perceptron, Data Mining can be applied to android authenticity prediction. These techniques can help capture complex patterns in the data that may be difficult to detect using traditional modeling approaches.
2. **Developing AI bases Real-time Approach -** AI-based approaches of real time Deep learning models, can be used to identify and examine new malicious applications. These methods can be used to recognize malicious apps by analyzing the app's short history, browsing activity, and code structure. Further research should also be

conducted to minimize online fraud and malware attacks by developing robust AI-based solutions.

3. **Cloud Computing for Huge Data -** Cloud computing can be used to store a large volume of Android data and detect malicious activities. So that we can get accurate predictions of the authenticity of the applications.

VIII. CONCLUSION

In this project, our goal was to use classification techniques to solve the problem of detecting malware applications on Android phones. We experimented with different supervised classification techniques and identified the best technique for each approach. The results obtained from this project show that artificial intelligence can be used effectively to detect and prevent malicious Android app downloads. This authenticator can save Android users from the potential dangers of downloading and installing malicious Android apps. By incorporating this AI-based authenticator into existing anti-malware safeguards, users can rest assured that their devices are protected from a wide variety of malicious threats. In addition, Random Forest and XGBoost Classifiers provide great results in the approaches.