**Android Applications Authenticity Prediction**

**Manisha Dhanuka**

**Data science trainee,**

**AlmaBetter, Bangalore**

**Abstract:**

Android is the most popular operating system in the world, with over 2.5 billion active users spanning over 190 countries. Created by ***Andy Rubin***, it quickly became the favorite OS for most of the mobile manufacturers in the early 2010’s. Being favorite of so many users, the statistics of having highest malware rate i.e., 47.15% of all vulnerable devices is a concern for android mobile users.

In this project, we develop a model using machine learning techniques that can differentiate between the malware apps and the benign ones.

1. **Problem Statement:**

The dataset consists of nearly 30,000 apps from various sources: Google's play store, hiapk, app China, Android, mumayi, gfan slideme, and pandaapp along with the extracted permissions with the package.

The main objective is to find on the basis of these features what apps are the problem creators for the Android users.

**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
* Dangerous Permissions: Total 74 permissions with the Boolean feature indicating their presence or absence.
* Safe Permissions: Total 99 permissions with indicating feature for each safe permission.
* Related apps: Apps related to installed App
* Price: Price of the app

1. **Introduction:**

Malware, contraction for “***Malicious Software***”, is intrusive software specially designed to disrupt or to damage a computer, server etc., or which unknowingly interferes with the user’s computer security and privacy.

1. **Types of Malwares:**

Some types of Malwares that infect Android are:

* **Trojan Horse:** As, the name suggests, it misrepresents itself as benign ones to persuade people to install it either through mails or messages and carries hidden destructive function when the application is started.
* **Virus:** A software that embeds itself in some other executable software on the target system without user’s knowledge and consent and when it is run, it spreads to other executable files.
* **Ransomware:** Type of malware that threatens to publish the victim’s personal data or perpetually block access to it unless a ransom is paid.
* **Adware:** It is designed to show frequent ads to a user in the form of pop-ups, sometimes redirecting you to web pages or applications
* **Banker Malware** attempts to steal users’ bank credentials without their knowledge

**4. How Malware Apps work?**

Cybercriminals looking to have a greater return focus their efforts on organizations and use a variety of tactics to infect the maximum number of corporate devices. Here are a few ways mobile malware can get to your device:

* Infected Applications
* Malvertising
* Scams or Phishing
* Direct to device

**5. Mobile malware prevention:**

The best remedy is prevention. Stay safe from malware by following this guidance:

* Don’t download apps from third-party app stores
* Read the reviews before downloading apps
* Install OS updates as soon as they are available to ensure important security patches are in place
* Check app permissions before installing an app in case it is asking to access something it doesn’t need to, such as camera access for a calendar app, for example.

**6. Steps Involved:**

* **Data Cleaning:**
* Handled the null values by imputing some of the values and dropping those values which can’t be imputed.
* Drop the duplicates to avoid bias before data modelling.
* Feature Selection that is removing the variables which were constant that is were not positively contributing to the modelling and the one which were not of that importance.
* **Exploratory Data Analysis:**

After Data Cleaning, I explored the data for better understanding the distribution of the features, their statistical values, relationship between them and with the class labels using ***matplotlib, seaborn and pandas*** library. During Analysis, we found that the apps with price more than zero are all malware apps that is it could be a great differentiator for this dataset but not valid for all. Thus, couldn’t use ***Decision Tree Classifier*** for modelling. Moreover, most of the features were not in linear relationship with the class Label, So ***Logistic Regression*** for the classification could not be used here. Also, the category columns show variations in their class label. So used that feature for the modelling.

* **Encoding Categorical Variables:**

As pointed out above, we used Category column for the modelling, so we had to convert them into Booleans using ***One Hot* *Encoder*** from ***sklearn library*** of python, which creates a unique feature per unique entry of the category column.

* **Splitting Data:**

Given problem was the supervised Classification problem. Thus, it was possible for us to evaluate the model. So, we split the data using ***Train- Test split*** so that we could check the performance of our model before finalizing the model on unseen data.

* **Feature Scaling:**

Some of the Algorithms which we were going to used were distance-based Algorithms and since the scale of the features were different. We have to bring them to equal level to give equal weightage. There are many Scaling Algorithms inbuilt in sklearn library like ***MinMax Scaler, Standard Scaler***. We used Standard Scaler which transform the continuous variables into Standard Normal Distribution.

* **Data Modelling:**

We then used some of the Classification Algorithms whose prerequisites were seem to be satisfied by the data. The Algorithms used are as:

* Random Forest
* Gradient Boosting
* Support Vector Machine
* K-Nearest Neighbors
* Naïve Bayes
* **Model Evaluation:**

We used various Classification metrics to evaluate the performance of our model like:

* Recall Score
* Precision Score
* Accuracy Score
* ROC- AUC Score
* **Cross- Validation and Hyper-Parameter Tuning:**

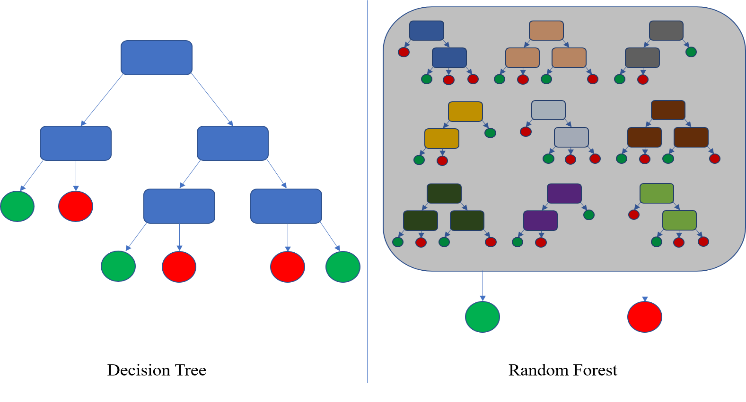
After analyzing the metrics, there were some models that were performing not good and some were giving similar performances. So, we did hyperparameter tuning on Random Forest, Gradient Boosting and KNN for better results. For this, we used ***GridSearchCV*** from the sklearn library.

* **Final Model:**

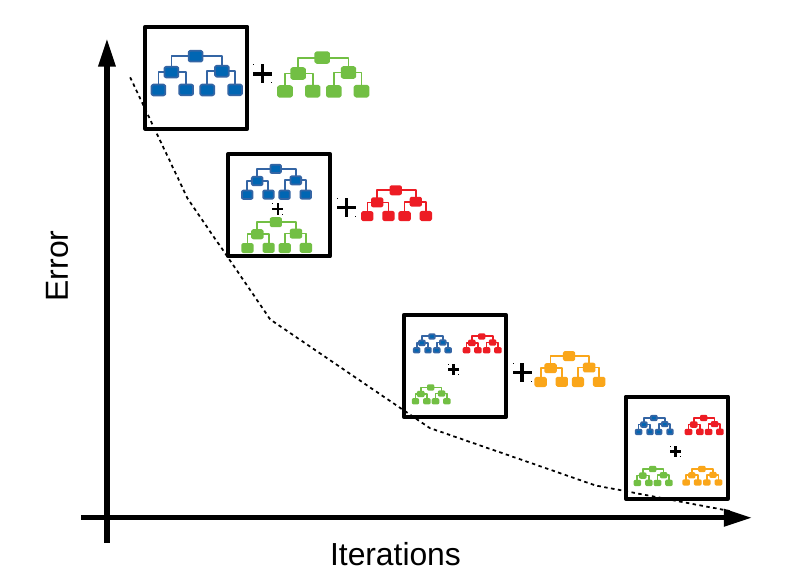
After all the hyperparameters tuning done, we find that Gradient Boosting was doing better. So, we created the final model with the best parameters GridSearchCV provided us and then fit the entire data on it.

**7. Classification Algorithms used:**

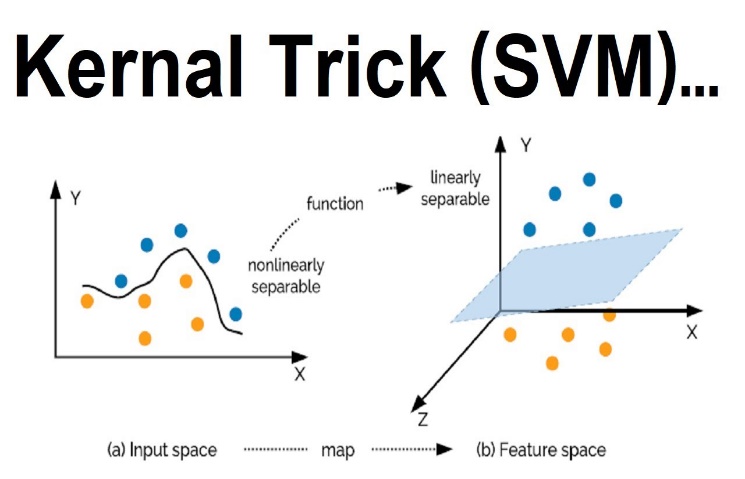
**1. Random Forest:** Random Forest is also called as “Ensemble Learner” since it relies on ensemble of models. In Tree Based Classifier, there could be a chance that all features are not used or root node is always same. Random Forest solve this problem by randomly picking features at each split and then taking the majority votes of all the trees.

****

**2. Gradient Boosting:** The main idea behind this algorithm is to build models sequentially and these subsequent models try to reduce the errors of the previous model. This is done by building a new model on the errors or residuals of the previous model.



**3. Support Vector Machine:** SVM is a non-probabilistic classifier. It builds a model that assign new data points to one of the given categories. It can handle both types: linear as non-linear classification using the concept of ‘Kernel Trick’. The key here is that for the fit only the position of support vectors i.e., the vectors at the border of the margin matters only. In short, they are insensitive to the exact behavior of distant points. Thus, can handle outliers.



**4. K-Nearest Neighbors:** It can be used for regression as well as classification. It works on the simple algorithm which assigns the label to the data according to the distance from old label. It considers the pre-defined k nearest neighbors to come out with the final label, hence the name. For a tie breaker, it uses the sequence of the neighbors in the train data. Since, it is a distance-based algorithm, scaling becomes a necessity for the accurate results.



**5. Naïve Bayes:** Based on Bayesian Classification Methods. It is extremely fast and simple algorithm and suitable for very high dimensional datasets. It tries to calculate the posterior probability P(Y|X) using prior probability P(Y) and likelihood P(X|Y). Thus, a kind of generative model. Also, for making it simple, it makes an assumption that the features are independent. Here we used GaussianNB which assumes that the features are from independent Normal Distributions.

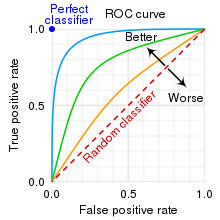
**8. Classification Metrics:**

**1. Accuracy:** It answers ‘*How often the model is correct?’.* It is the ratio of correct predictions to the total values. But it does not work well if there is a class imbalance – termed as “Accuracy Paradox”, which means if you have 97:3 class label ratio, a model which always predict the majority class will be given 97% accuracy but couldn’t be considered as a good model. Thus, one should be cautious while choosing it as the decisive metric.

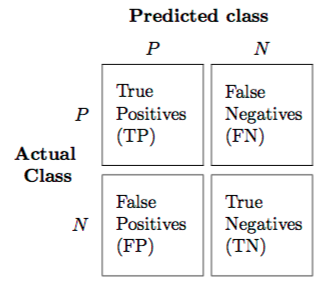
**2. Recall:** It answers *‘When actually it is positive, how often is it correct?*’. It is the ratio of True positives to the total actual positives.

**3. Precision:** It answers ‘*When prediction is positive, how often is it correct?’*. It is the ratio of True positives to the predictive positives.

**4. ROC-AUC Score:** It gives the area under Receiver Operator Characteristic Curve (ROC). ROC curves use a combination of the true positive rate (the proportion of positive examples predicted correctly, defined exactly as recall) and false positive rate (the proportion of negative examples predicted incorrectly) to build up a summary picture of the classification performance.



**5. Confusion Matrix:** The confusion matrix is a table that summarizes how successful the classification model is at predicting examples belonging to various classes.

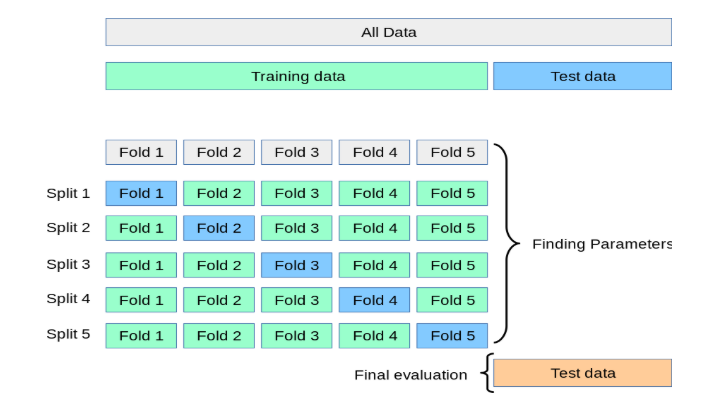


**9. Hyperparameter Tuning and Cross- Validation:** Parameters which define the model architecture are referred to as hyperparameters and thus this process of searching for the ideal model architecture is referred to as hyperparameter tuning. The model parameters specify how to transform the input data into the desired output, the hyperparameters define how our model is actually structured.

Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations.

**Grid SearchCV:**

Grid search is arguably the most basic hyperparameter tuning and cross-validation method. With this technique, we simply build a model for each possible combination of all of the hyperparameter values provided, evaluating each model, and selecting the architecture which produces the best results.



**Conclusion:**

After doing all above-described steps we observed that:

Random Forest was overfitting the model.

Naïve Bayes was not working that well, maybe data doesn’t fit into its assumptions.

After Hyperparameter Tuning, Gradient Boosting Algorithm was giving the better results, with 84% recall, 89% precision and 83 % Accuracy. Thus, we created our final model with the maximum depth as 5 and n\_estimators as 10.