**ANDRIOD AUTHENTICITY PREDICTION**

**Arunav Goswami, Nayanjyoti Sharma, Mohammed Saad Pasha**

**Data science trainees,**

**AlmaBetter, Bangalore**

**Abstract:**

In recent years, smartphones have brought people’s lives to a new high level. Smartphone applications, or Apps, are accelerating the process with many more functions getting developed, such as browsing the Internet, making payments, taking photos and share. However, the "Apps" are bringing potential vulnerability when they access private information from the phones, and mobile security has never been so much focused on like today.

We used different machine learning models like Logistic regression, Decision Tress, Random Forest, Gradient Boosting, XGBoost, KNN and Naïve bayes using cross validation approach for predicting whether the app is benign or malware. Multiple evaluation indices such as Precision, Recall, F1-Score, ROC-AUC Score are used to measure the prediction performance of the classification models.

**1. Problem Statement**

This dataset consists of apps needed permissions during installation and run-time. We collect apps from three different sources google play, third-party apps and malware dataset. This file contains more than 30,000 Android apps. features extracted at the time of installation and execution. One file contains the name of the features and others contain .apk file corresponding to it extracted permissions with respective package. Apps are collected from Google's play store, hiapk, app china, Android, mumayi , gfan slideme, and pandaapp. These .apk files collected from the last three years continuously and contain 81 distinct malware families. But, Here you are only supposed to predict whether the app is benign (0) or malware(1).

We have the following variables in the dataset:

Class :- Whether the app is Benign(0) or Malware(1) :-

App :- Name of the App

Package :- OBB/Data package installed in root folder

Category :- App Category (eg. 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

Default: Access DRM content. (S) :- 0 : No , 1 : Yes

Phone calls: modify phone state (S) :- 0 : No , 1 : Yes

We have shown only a part of the features since there are 184 total features.

**2. Introduction**

For enforcing security, Android platform uses authorizing system which grants permission per application at install-time. With authorized privilege, user applications can modify and delete user's personal information. Therefore, inspection of granted permission usage can be used to detect security vulnerabilities. ISO/IEC 25 010 defines software product security characteristic and provides guidelines to evaluate software product quality. Among sub-characteristics of security, Authenticity is related to Android permission system. In this paper, we present authenticity metric for android application. This metric can quantify the permission usage of application and measured information can be used to classify the malware applications.

**3**. **Steps involved:**

i)Installing libraries and gathering the dataset

i) Pre-processing the dataset: - Checking for Missing values, Duplicate values etc.

ii) Exploratory Data Analysis: - Analysing the dependent variable, categorical and numerical variables individually.

iii) Feature Engineering: - Creation of new features according to our need, dropping of unnecessary data points or features by checking correlation, VIF etc., handling of outliers, One–Hot Encoding and normalization of features.

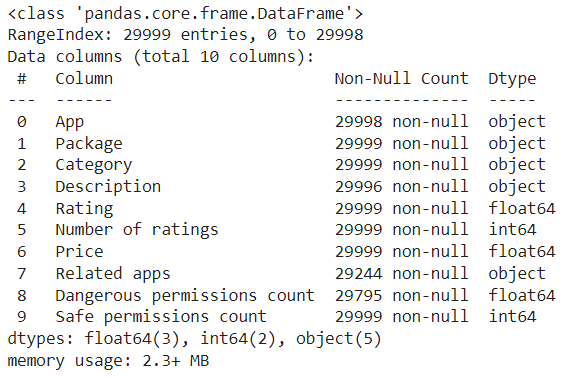
iv) Fitting of Machine Learning models with training dataset and evaluating with testing dataset: - like Linear regression, Decision Tress, Random Forest, Gradient Boosting

v) Hypertuning and Explaining the best model with LIME and ELI5.

**4. Pre-processing:**

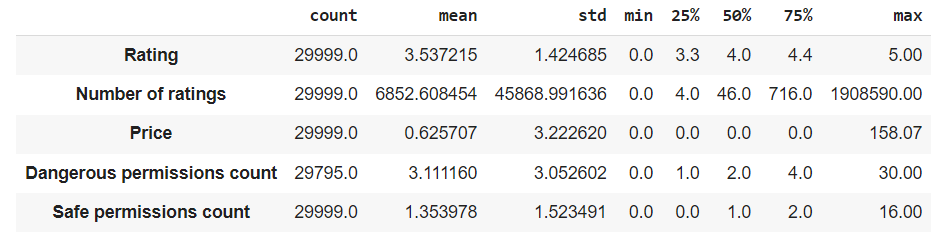
**4.A. Summary of the dataset**

After loading the dataset, we first check a concise summary of the data frame:

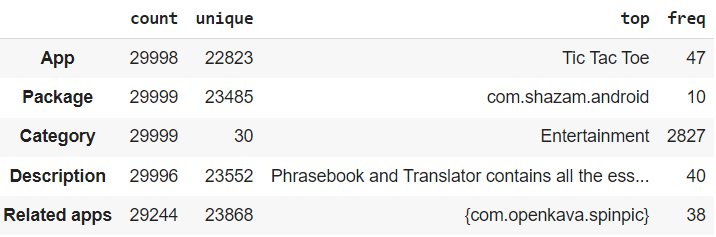


This summary is only for 10 features. There are total 184 features and 29999 rows.

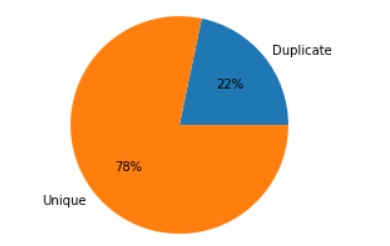
**Data Summary for Numerical Features:**



**Data Summary for Categorical Features:**

****

**4.B. Checking Duplicate Values:**

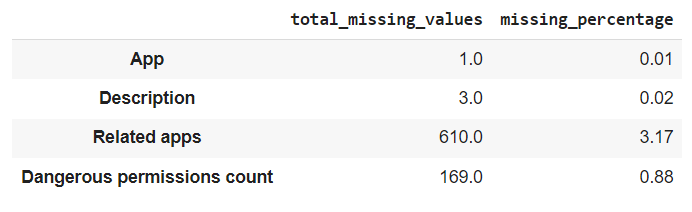
****

In android, app names can be same but the package names must be different. So, we have checked the duplicate rows for the Package column.

From the pie chart, we see that the dataset has only 78% unique apps. The other 22% are duplicate entries. We have found that there is a total of 6514 rows which have duplicate entries, thus we drop them.

Then we dropped the ‘Package’ feature as it is unique.

**4.C. Checking Missing Values:**

****

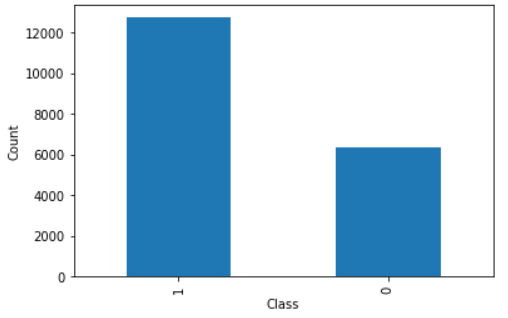
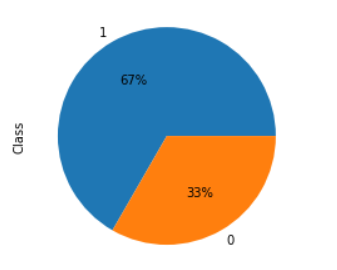
Prediction of an app whether it is benign(0) or malware(1) doesn't depend on the app name and it's description columns. So, we can remove these columns.

The "Related apps" columns represent similar apps (e.g. same category). This column is unimportant for our problem and thus remove it.

The "Dangerous permissions count" column has less than 1% missing values. So, we can drop these rows.

**5. EDA and Feature Engineering:**

**5.A. Analysing Dependent Variable ‘Class’:**

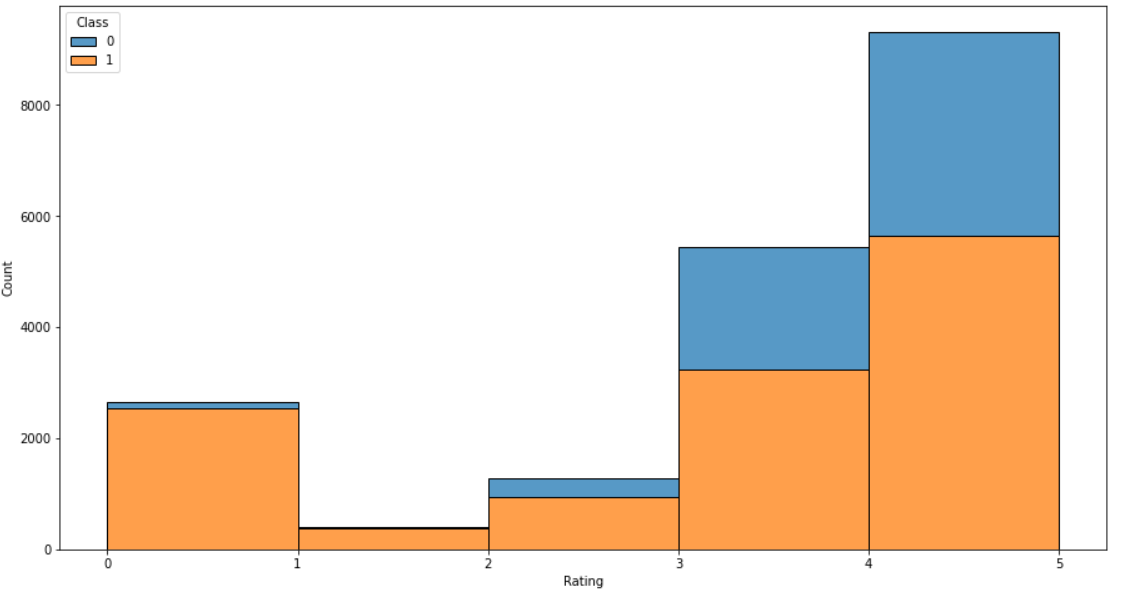
** **

67% apps are malware and rest 33% are Benign in total.

5.B **Analysing Independent Variables:**

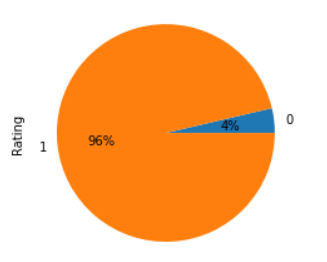
**i) Rating:**

Plotting 'Rating' and 'Class':



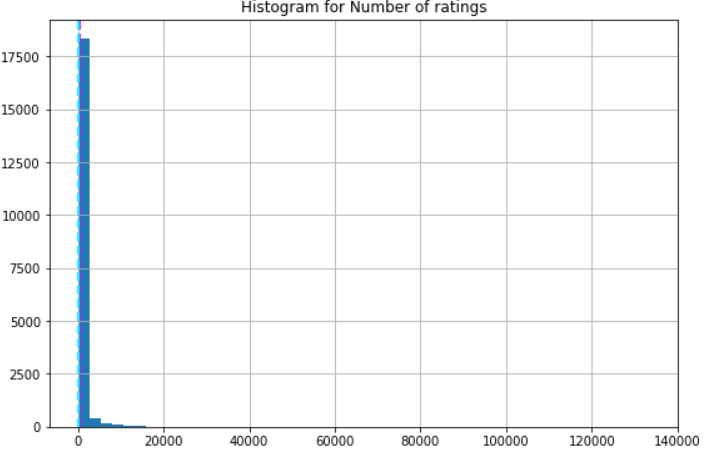
* Between 0 to 3 a most of the apps have malware.
* From 3 to 5 there are more benign apps compared to ratings between 0-3.

We then analysed the zero rated apps with our Class variable.

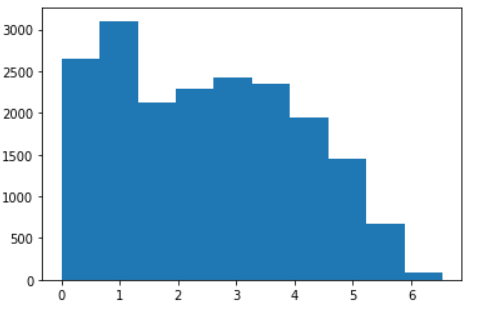


We found that 96% apps are malware if it has 0 rating.

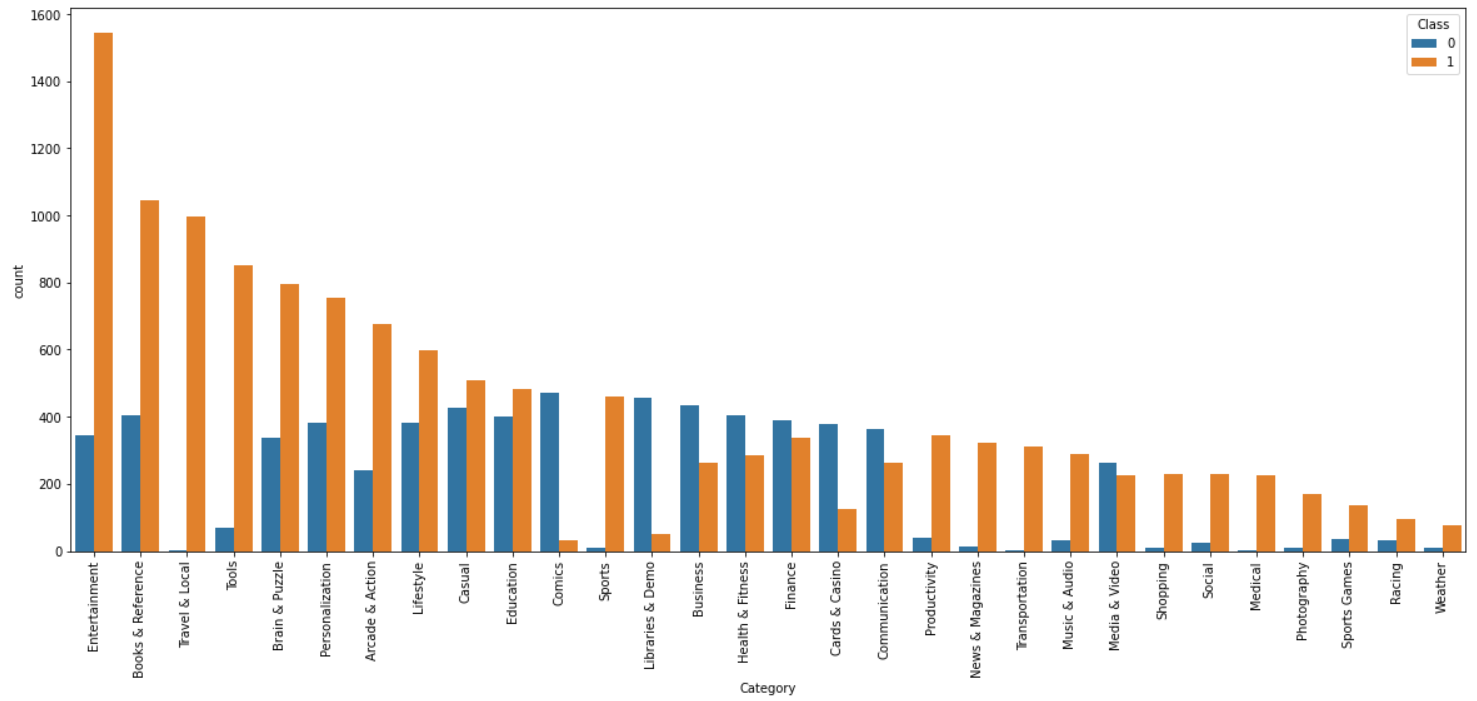
**ii) Number of ratings:**

****

Number of ratings' is extremely positive skewed. So, We have used 'boxcox' method to transform the ‘Number of ratings’ column to normally distributed.

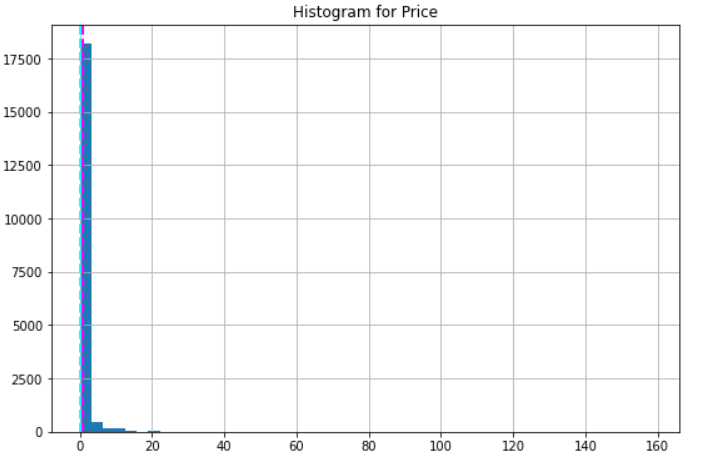


**iii) Category**



For the categories, ‘Travel & Local’, ‘Tools’, ‘Sports’ etc., almost all apps are malware. For the categories, ‘Comics’, ‘Libraries & Demo’ etc, almost all apps are benign.

**iv) Price**

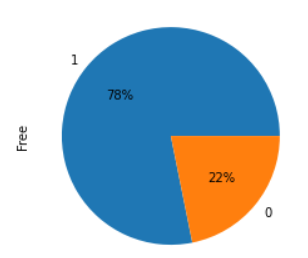
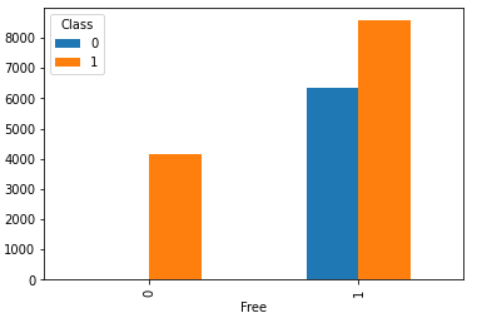
****

Most of the apps are priced between 0 to 20.

Now from ‘Price’ column, we have created a new column as ‘Free’.

**v) Free**

The app is Free if its price is 0.

We observe the following points from the charts:

i) Most of the apps in the dataset (78%) are Free.

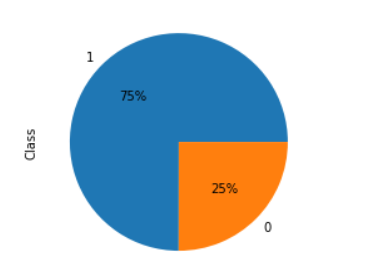
ii) All paid apps are malware.

iii) In Free apps, the chance of it being malware is higher than benign.

vi) **Dangerous permissions count**

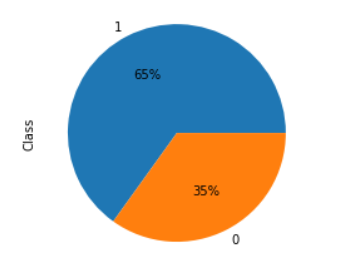
We have divided the dataset based on whether the app has dangerous permission or not:

a) App without Dangerous permissions (= 0):



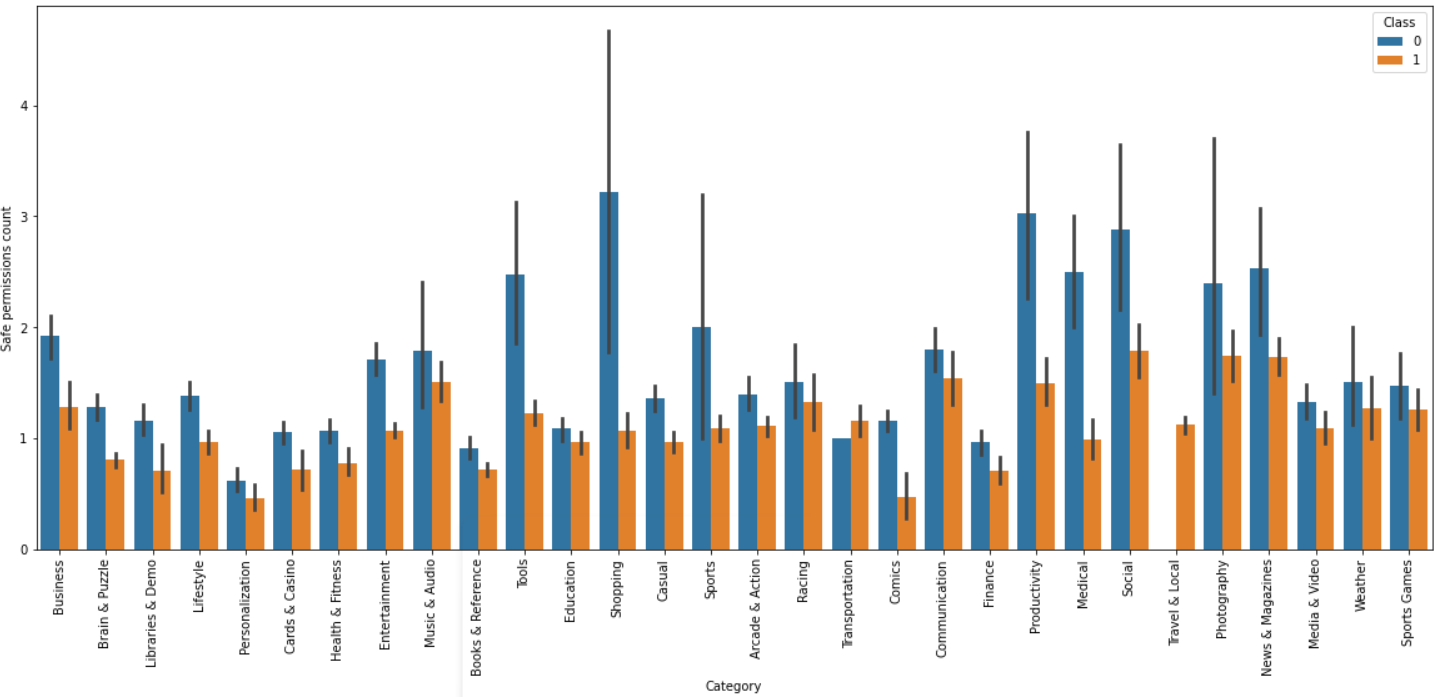
75% of the apps which does not require dangerous permissions are malware.

b) App with Dangerous permissions (> 0):



65% of the apps which require dangerous permissions are malware.

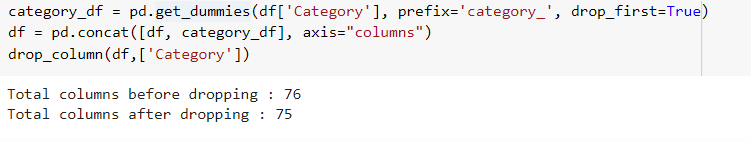
vii) **Safe permissions count**



Most of the apps which require safe permission count are Benign apps except for ‘Travel & Local’ and ‘Transportation’ Category.

**5.C Encoding:**

There is one categorical feature i.e. ‘Category’ where we have used one hot encoding to transform it into numerical values.

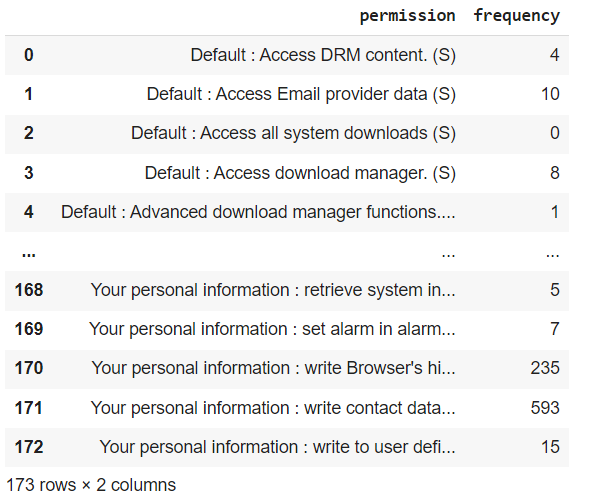


After encoding, original category column is dropped.

**6 Feature Selection: -**

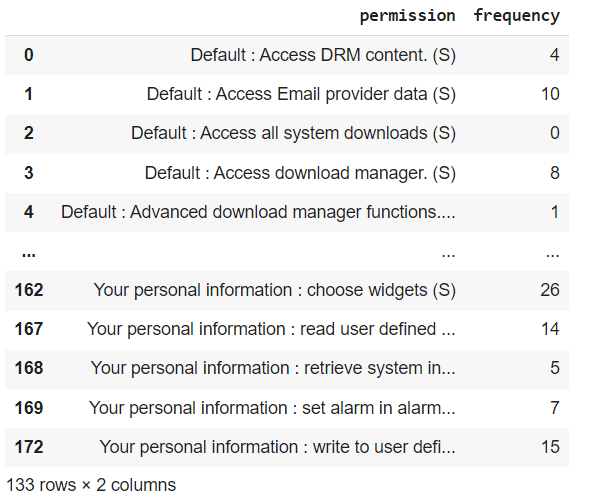
**6.A. Basic filtering:**

**Finding All permission columns - all columns containing android permissions**

****

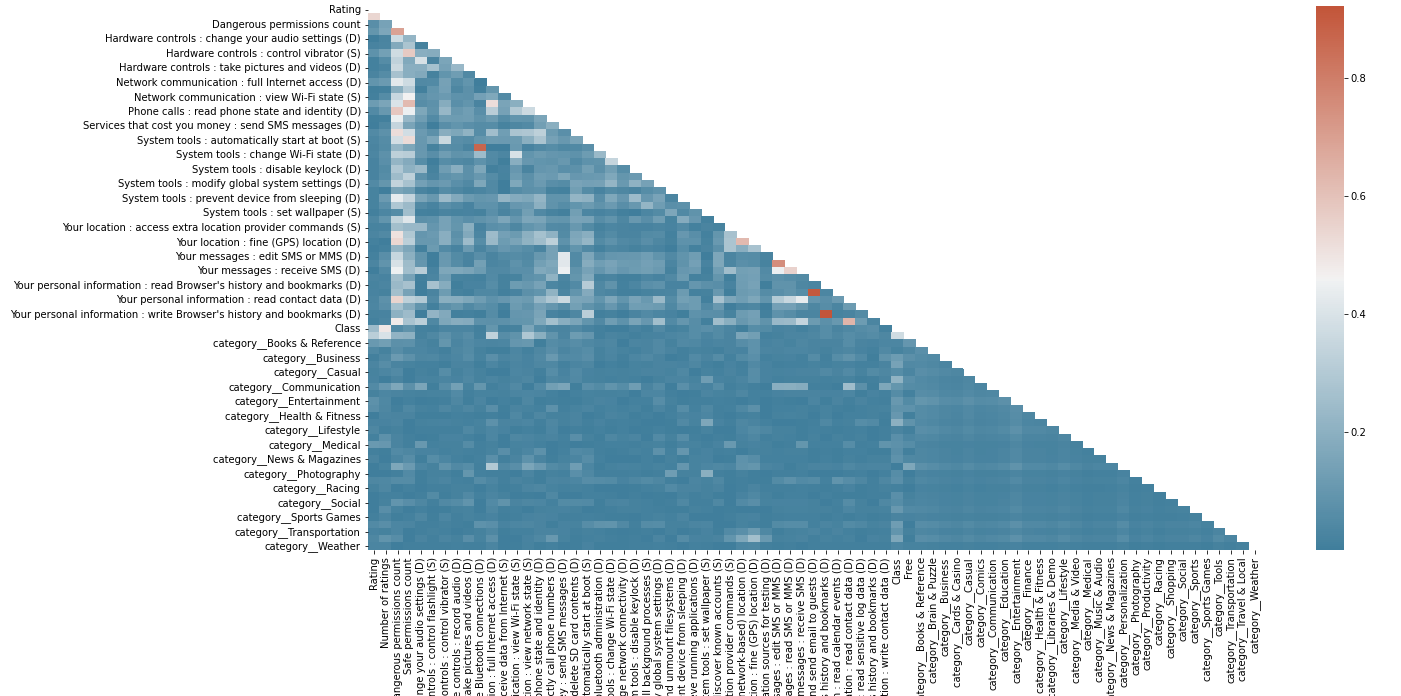
There are total 173 columns that are related to different android permissions. We calculated the frequency of each.

**We have found permissions which are rarely used in apps. In our case, we have found permissions which are used in less than 201 apps(~1% of total apps).**

****

So after filtering, we found that there is a total of 133 permissions that are rarely used. We drop these permission features from our data.

**6.B. Correlation**



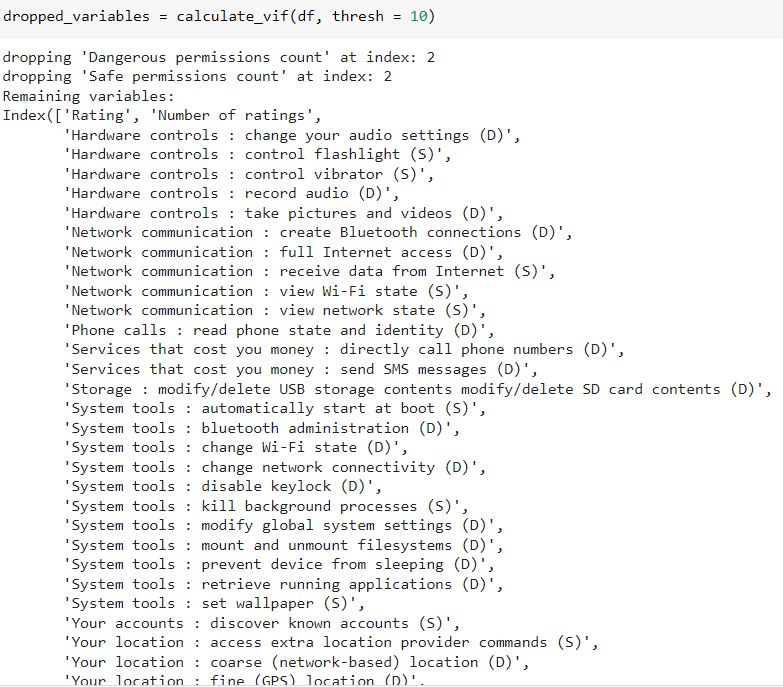
Correlated Features above 90% are dropped:

i) 'Your personal information: read calendar events (D)‘ and

ii) ‘Your personal information: write Browser's history and bookmarks (D)’

6.C. **Variance Inflation Factor (VIF):**

Variance inflation factor (VIF) is a measure of the amount of multicollinearity in a set of multiple regression variables. Values of VIF that exceed 10 are often regarded as indicating multicollinearity.



Here we have used automated function to drop the features with VIF value > 10

Dropped columns are:

i) 'Dangerous permissions count’ and

ii) 'Safe permissions count'

**7. A Train Test split:**



The dataset is split into the Training set and Test set in 70:30 ratio.

**7.B. Transformation of features:**

We have used the MinMaxScaler technique to transform features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.

**7.C. Algorithms:**

We will use various regression models to predict our target variable i.e. ‘Rented Bike Count per hour’:

**i) Logistic Regression**: Logistics regression uses sigmoid function above to return the probability of a label. It is widely used when the classification problem is binary — true or false, win or lose, positive or negative. The sigmoid function generates a probability output. By comparing the probability with a pre-defined threshold, the object is assigned to a label accordingly.

**ii) Decision Tree:**

Decision Tree regression builds a tree-like structure in which each internal node represents the "test" for an attribute, each branch represent the result of the test, and each leaf node represents the final decision or result. Decision trees are good at capturing non-linear interaction between the features and the target variable.

**iii) Random forest:**

The Random Forest regression is an ensemble learning method which combines multiple decision trees and predicts the final output based on the average of each tree output. The combined decision trees are called as base models. Random forest uses Bagging or Bootstrap Aggregation technique of ensemble learning in which aggregated decision tree runs in parallel and do not interact with each other.

**iv) Gradient Boosting:**

It utilizes a gradient descent algorithm that can optimize any differentiable loss function. An ensemble of trees is constructed individually, and individual trees are summed successively. The next tree tries to restore the loss (difference between actual and predicted values).

**v) KNN:**

K-Nearest Neighbor (KNN) algorithm predicts based on the specified number (k) of the nearest neighbouring data points. Here, the pre-processing of the data is significant as it impacts the distance measurements directly.

**vi)** **Naive Bayes**:

Naive Bayes is based on Bayes’ Theorem — an approach to calculate conditional probability based on prior knowledge, and the naive assumption that each feature is independent to each other. The biggest advantage of Naive Bayes is that, while most machine learning algorithms rely on large amount of training data, it performs relatively well even when the training data size is small.

**vii)** **XGBoost**:

XGBoost is one of the fastest implementations of gradient boosting. trees. It does this by tackling one of the major inefficiencies of gradient boosted trees: considering the potential loss for all possible splits to create a new branch .XGBoost tackles this inefficiency by looking at the distribution of features across all data points in a leaf and using this information to reduce the search space of possible feature splits.

**8. Model Evaluation Metrics:**

We will use following classification model evaluation metrics to check the performance of our machine learning model:

**1. Confusion Matrix-**

The confusion matrix is a table that summarizes how successful the classification modelis at predicting examples belonging to various classes. One axis of the confusion matrix is the label that the model predicted, and the other axis is the actual label.

**2. Precision/Recall-**

Precision is the ratio of correct positive predictions to the overall number of positive predictions : TP/TP+FP

Recall is the ratio of correct positive predictions to the overall number of positive examples in the set: TP/FN+TP

**3. Accuracy-**

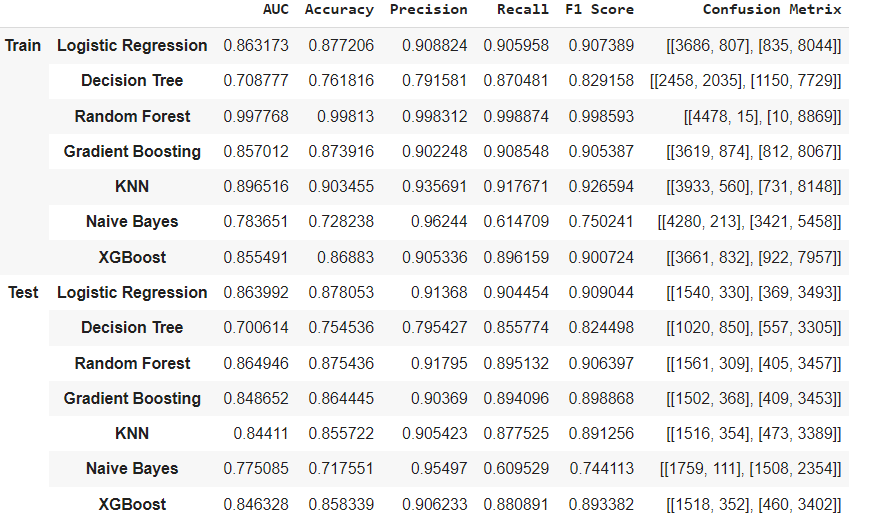
Accuracy is given by the number of correctly classified examples divided by the total number

of classified examples. In terms of the confusion matrix, it is given by: TP+TN/TP+TN+FP+FN

**4.** **Area under ROC Curve(AUC)**-

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.

After applying machine learning models and calculating evaluation metrics for each model, we find results as follows: -



Observation: Logistic Regression has the highest F1 Score for testing dataset.

We use F1 score since our dataset is slightly imbalanced and there is a serious downside to predicting false negatives. Among all models, Logistic Regression has the best F1 Score of almost 91% for both train and test dataset. . So, we will select this model and find the best hyper parameters for it.

**9.** **Hyper parameter Tuning and Cross Validation:**

Hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a model argument whose value is set before the learning process begins.

**i)** **Solver:** Algorithm to use in the optimization problem. We have used ‘saga’ since our dataset is large.

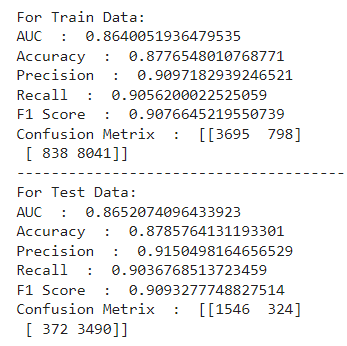
**ii) Penalty: Regularization method - ‘l1’, ‘l2’, ‘elasticnet’**

**iii) C: The C parameter controls the penalty strength**

**Method used from Hyperparameter tuning:**

**RandomizedSearchCV**:- Random search is a technique where random combinations of the hyperparameters are used to find the best solution for the built model. It is good in testing a wide range of values and normally it reaches a very good combination very fast.

After Applying Hyperparameter tuning on Logistic Regression, we observed that the performance metrics increased slightly:

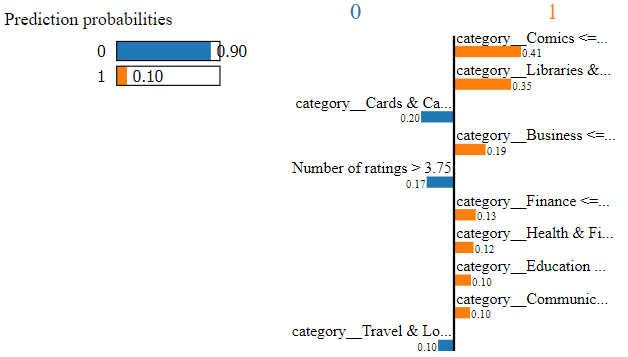
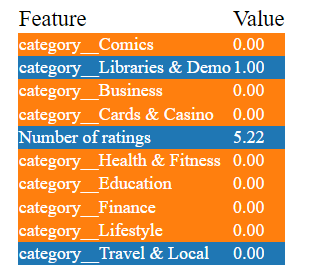


**10.** **Model Explainability:**

Model explainability refers to the concept of being able to understand the machine learning model. Once we understand a model, we can detect if there is any bias present in the model. Model Explainability becomes important while debugging a model during the development phase.

**LIME:**

Local Interpretable Model-Agnostic Explanations (LIME) explains the predictions of any classifier in “an interpretable and faithful manner, by learning an interpretable model locally around the prediction.”

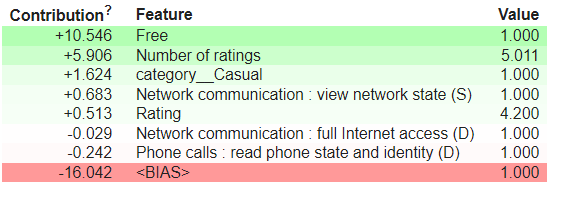
 

The idea behind LIME is to provide the reasons why a prediction was made. We create the LIME plot by taking a datapoint from the test. From the LIME plot we can conclude that Category Cards & Casino, Number of ratings and category Travel & Local mainly contribute for the app to be benign.

**ELI 5:**

ELI5 is an acronym for ‘Explain like I am a 5-year old’. Python has ELI5 methods to show the functionality for both:

* Global interpretation-Look at a model’s parameters and figure out at a global level how the model works
* Local interpretation-Look at a single prediction and identify features leading to that prediction.



ELI5 gives the most important feature and its contribution. From the above plot we can see that the features 'Free', 'Number of ratings' are most important feature for the prediction.

**11. Conclusion:**

* 22 % rows consist of duplicate values.
* Given dataset is slightly imbalanced because 67% apps are malware and rest 33% are Benign.
* Between Rating 0 to 3, most of the apps have malware. From 3 to 5, there are more benign apps as compared to ratings between 0-3.
* For the categories, ‘Travel & Local’, ‘Tools’, ‘Sports’ etc., almost all apps are malware. For the categories, ‘Comics’, ‘Libraries & Demo’ etc, almost all apps are benign.
* All paid apps are malware and number of malware apps is higher than benign in the free apps. But it does not make sense for all paid apps are to be malware. It may be due to misclassification of apps.
* We use F1 score since our dataset is slightly imbalanced and there is a serious downside to predicting false negatives. Among all models, Logistic Regression has the best F1 Score of almost 91% for both train and test dataset.