Google Store App Rating Prediction

Ensemble Techniques - Graded Project

Ashwath J

1. Install the necessary libraries and read the provided dataset. (1 point)

```
In [50]: import numpy as np
         import pandas as pd
         import seaborn as sns
         import matplotlib.pyplot as plt
```

2. EDA and Preprocessing (27 points)

a. Check the info and summary statistics of the dataset. List out the columns that need to be worked upon for model building. (2 points)

```
df['Rating'].isna().value_counts()
```

```
False    9367
True     1474
Name: Rating, dtype: int64
```

```
df['Rating']=df['Rating'].fillna(df['Rating'].mean().round(1))
```

```
df['Rating'].isna()
```

```
0        False
1        False
2        False
3        False
4        False
         ...
10836    False
10837    False
10838    False
10839    False
10840    False
Name: Rating, Length: 10841, dtype: bool
```

The 1st important step is to impute empty Rating rows.

Google Store App Rating Prediction

Ensemble Techniques - Graded Project

Ashwath J

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   App             10841 non-null  object
 1   Category        10841 non-null  object
 2   Rating          10841 non-null  float64
 3   Reviews         10841 non-null  object
 4   Size            10841 non-null  object
 5   Installs        10841 non-null  object
 6   Type            10840 non-null  object
 7   Price           10841 non-null  object
 8   Content Rating  10840 non-null  object
 9   Genres          10841 non-null  object
 10  Last Updated    10841 non-null  object
 11  Current Ver     10833 non-null  object
 12  Android Ver     10838 non-null  object
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

```
df.describe()
```

	Rating	Reviews	Size	Price	Category_1.9	Category_ART_AND_DESIGN	Category_AUTO_AND_VEHICLES	Category_BEAUTY
count	9660.000000	9.659000e+03	9345.000000	9659.000000	9660.000000	9660.000000	9660.000000	9660.000000
mean	4.178810	2.166512e+05	18.389984	1.097231	0.000104	0.006315	0.008799	0.005487
std	0.516558	1.830738e+06	21.613483	16.851618	0.010174	0.079218	0.093395	0.073872
min	1.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	4.000000	2.500000e+01	3.300000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	4.200000	9.690000e+02	9.800000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	4.500000	2.940100e+04	26.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	19.000000	7.812821e+07	100.000000	400.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 188 columns

b. Check if there are any duplicate entries for the apps (1 point)

droping duplicate rows

```
df.drop_duplicates('App', keep = 'last', inplace = True)
```

```
df.drop(df[df['Category']=='1.9'].index)
```

Ashwath J

C. Check if there are any wrong values in the ‘Category’ column and impute them with relevant values. (2 points)

```
col = X.columns
for i in col:
    X[i].fillna(X[i].median() ,inplace = True)
```

Filling all nas with their corresponding column medians.

d. Which category has the highest number of apps? (2 points)

```
df["Category"].value_counts().sort_values
```

```
<bound method Series.sort_values of FAMILY
GAME                    1144
TOOLS                    843
MEDICAL                  463
BUSINESS                 460
```

Game category has the highest number of apps.

e. Check the distribution of rating column and convert ratings into two categories and save it in the data frame as ‘Rating_cat’ (high = +>3.5 and remaining as low) (2 points)

```
ash = []
for i in y:
    if(i<3.5):
        a = "low"
        ash.append(a)
    else:
        a = "High"
        ash.append(a)
```

Rating is converted into a binomial category column.

f. Convert the ‘Review’ column to a numerical column and impute invalid values if there are any

Google Store App Rating Prediction

Ensemble Techniques - Graded Project

Ashwath J

2F) Reviews to numeric

```
df['Reviews']=pd.to_numeric(df['Reviews'],errors='coerce')
```

```
df
```

h. Make the values of 'Size' as integers by replacing M and K with correct values. Convert all the values to numeric and make invalid values to NaN. (3 points)

2H) removing M and k from df

```
df['Size']=df['Size'].apply(lambda x:x.replace('M','').replace('Varies with device','0'))
df['Size'] = pd.to_numeric(df['Size'], errors = 'coerce')
```

n. Encode categorical column (Type, Rating_categories, Category) [Hint - use get_dummies] (2 points)

2n) Encoding Cat Columns

```
oneHotCols = ['Category','Installs','Type',"Content Rating",'Genres']
df=pd.get_dummies(df, columns = oneHotCols)
```

```
df["Category"].value_counts().sort_values
```

```
<bound method Series.sort_values of FAMILY                1972
GAME                 1144
TOOLS                 843
MEDICAL               463
BUSINESS              460
PRODUCTIVITY          424
PERSONALIZATION       392
COMMUNICATION         387
SPORTS                384
LIFESTYLE             382
```

m. Drop columns that you feel can not be used for model building. ExampleApp, Content Rating, Genre, Last updated, Current Ver, and Android Ver columns from the final data frame. (2 points)

2M) Droping columns which is not important for model building

```
X=df.drop(labels=['Rating','App','Reviews',"Last Updated","Current Ver","Android Ver"],axis=1)
y=df['Rating']
```

Google Store App Rating Prediction

Ensemble Techniques - Graded Project

Ashwath J

k. Remove the “$” sign the “Price” column values and make it a numerical column. (2 points)

2K) removing $ symbol from price

```
df['Price']=df['Price'].apply(lambda x:x.replace('$',''))
df['Price'] = pd.to_numeric(df['Price'], errors = 'coerce')
```

```
df.dtypes
```

```
App                object
Category           object
Rating            float64
Reviews           float64
Size               object
Installs           object
Type               object
Price             float64
Content Rating     object
Genres             object
Last Updated       object
Current Ver        object
Android Ver        object
dtype: object
```

3. Prepare data for modeling. (2 points) a. Segregate dependent variable and independent features into two separate variables and split the data into train and test set [Use 70:30 split]

Prepare data for modeling

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, ash, test_size=.30, random_state=1)
```

4. Build a classifier model to predict the rating category (Rating_cat - high or low) using the following algorithm and make predictions on the test data. Evaluate the model and report your results. (16 points - 4 points each) a. Decision Tree Classifier b. Random Forest model c. Gradient Boosting model d. Stacking model

Google Store App Rating Prediction

Ensemble Techniques - Graded Project

Ashwath J

DECISION TREE CLASSIFIER

```
from sklearn.tree import DecisionTreeClassifier

dTree = DecisionTreeClassifier(criterion = 'gini', random_state=1,max_depth = 3)
dTree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(max_depth=3, random_state=1)
```

```
print(dTree.score(X_train, y_train))
print(dTree.score(X_test, y_test))
```

```
0.9219165927240461
0.9347826086956522
```

RANDOM FOREST CLASSIFIER

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=50,random_state = 1,max_features = 10,max_depth=3)
rfc.fit(X_train,y_train)
rfc.score(X_test,y_test)
```

```
0.9361628709454797
```

GRADIENT BOOSTING MODEL

```
from sklearn.ensemble import GradientBoostingClassifier
gbc = GradientBoostingClassifier(n_estimators=50,random_state = 1,max_depth=2)
gbc.fit(X_train,y_train)
gbc.score(X_test,y_test)
```

```
0.9361628709454797
```

Google Store App Rating Prediction

Ensemble Techniques - Graded Project

Ashwath J

5. Check the importance of different features by using model.feature_importances_ function in Python (2 points)

```
pd.DataFrame(dTree.feature_importances_,columns=["Imp"],index=X_train.columns)
```

	Imp
Size	0.051183
Price	0.000000
Category_1.9	0.000000
Category_ART_AND_DESIGN	0.000000
Category_AUTO_AND_VEHICLES	0.000000
...	...
Genres_Trivia;Education	0.000000
Genres_Video Players & Editors	0.000000
Genres_Video Players & Editors;Music & Video	0.000000
Genres_Weather	0.000000
Genres_Word	0.000000

6. Comment on your results and findings from the above analysis. What can you infer about how to make a highly rated mobile App from this project? (2 points)

After undergoing these algorithms and processes, we concluded that our hypothesis is true. Meaning you can predict the app ratings, however significant preprocessing must be done before you start the classification and regression processes.The Play Store apps data has enormous potential to drive app-making businesses to success. Actionable insights can be drawn for developers to work on and capture the Android market! This shows that given the Size, Type, Price, Content Rating, and Genre of an app, we can predict about 91% accuracy if an app will have more than 100,000 installs and be a hit on the Google Play Store.