**App Rating Prediction (GooglePlayStore)**

# importing essential libraries

import numpy as np

import pandas as pd

# 1. Load the data file using pandas

PlayStoreData = pd.read\_csv('G:\\DS----PYTHON\\PYTHON ---- MAY ---08\\PROJECT --- PYTHON\\1569582940\_googleplaystore\\googleplaystore.csv')

PlayStoreData.head()

# to know for how many rows and columns are there in our dataset

PlayStoreData.shape

# We got 13 columns and 10841 rows/records

# (10841, 13)

## Check the datatype of each column present in our dataset using info function

PlayStoreData.info()

## Check the datatype of each column present in our dataset using info function

PlayStoreData.info()

3. Drop records with nulls in any of the columns.

# We will drop records where null values are present using dropna() function

# dropna() is used for dropping those NaN present records

​PlayStoreData.dropna(inplace=True)

# This will remove those records containing null values

# We can check again where null values records removed or not using

PlayStoreData.isna().sum()

### 1.Extract the numeric value from the column

### 2. Multiply the value by 1,000, if size is mentioned in Mb

# Check for the values which are present in the Size column using unique() function

PlayStoreData['Size'].unique()

# Define a function for removing 'M' and 'k' from the dataset

# multiply by 1000 in place of M contains column

def sizes(x):

if 'M' in x:

return float(x.replace("M",""))\*1000

if 'k' in x:

return float(x.replace('k',''))

return 0.0

# applying the function with the column to know whether M and K removed or not

PlayStoreData['Size']=PlayStoreData['Size'].apply(sizes)

PlayStoreData['Size']

# # we have successfully removed 'M' and 'k' sign and converted into float datatype from string datatype

PlayStoreData['Reviews']=PlayStoreData['Reviews'].astype(float)

PlayStoreData['Reviews']

# Name: Reviews, Length: 9360, dtype: float64

### 1.Treat 1,000,000+ as 1,000,000

### 2. remove ‘+’, ‘,’ from the field, convert it to integer

def pluss(x):

if '+' in x:

return x.replace('+','')

return 0.0

def coma(x):

if ',' in x:

return float(x.replace(',',''))

return 0.0# removing '$' sign from price column

def dollar(x):

if '$' in x:

return float(x.replace('$',''))

return 0.0

# 1. Average rating should be between 1 and 5 as only these values are allowed on the play store.

# Drop the rows that have a value outside following range.

# [(PlayStoreData['Rating']<1) & (PlayStoreData['Rating']>5)]

PlayStoreData[(PlayStoreData['Rating']<1) & (PlayStoreData['Rating']>5)].round(1)

# We found NO such records are there in our dataset in the range

# between [(PlayStoreData['Rating']<1) & (PlayStoreData['Rating']>5)]

# 2. Reviews should not be more than installs as only those who installed can review the app.

# If there are any such records, drop them.

print(len(PlayStoreData['Reviews']))

print(len(PlayStoreData['Installs']))

# There are equal length of Reviews and Install column

# i.e. those who install the app can only able to review the app

# 3. For free apps (type = “Free”), the price should not be >0. Drop any such rows.

# [(PlayStoreData['Type']=='Free') & (PlayStoreData['Price']>0)]

PlayStoreData[(PlayStoreData['Type']=='Free') & (PlayStoreData['Price']>0)]

# There are NO such apps which is free and price is greater than 0

# Range between [(PlayStoreData['Type']=='Free') & (PlayStoreData['Price']>0)]

# importing Visualization libraries

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import warnings

warnings.filterwarnings("ignore")

#Boxplot for Price

# Creating a boxplot for checking whether outliers are present or not in the column

sns.set\_style(style='whitegrid')

sns.boxplot(PlayStoreData['Price'])

plt.title('Prices of different apps', fontsize=18)

plt.xlabel('Price', fontsize=14)

# We can see that there are many outliers in Price column

# Creating a boxplot for checking whether outliers are present or not in the column

sns.boxplot(PlayStoreData['Reviews'])

plt.title('Reviews of different apps', fontsize=18)

plt.xlabel('Reviews', fontsize=14)

# We can see that there are many outliers in Price column

#Histogram for Rating

#How are the ratings distributed? Is it more toward higher ratings?

sns.histplot(PlayStoreData['Rating'],color='purple')

plt.title('Ratings of different apps', fontsize=18)

plt.xlabel('Rating', fontsize=14)

plt.ylabel('Frequency', fontsize=14)

# Rating is distributed as left skewed which we can say as Negetively skewed and it is towards higher ratings

# It is looking a very good distribution as left skewed

#Histogram for Size

sns.histplot(PlayStoreData['Size'], palette='Blues\_r')

plt.title('Sizes of different apps', fontsize=18)

plt.xlabel('Size', fontsize=14)

plt.ylabel('Frequency', fontsize=14)

# we can see that the distribution is a right skewed type and it starts from higher Size and continue to lower Size

#Drop these as most seem to be junk apps

# check for records which are greater than 200

PlayStoreData1=PlayStoreData[PlayStoreData['Price']>=200]

PlayStoreData1

# Assign that range in a new variable and get records for above mentioned condition

# Now we will drop records which price is greater than 200

PlayStoreData=PlayStoreData.drop(PlayStoreData1.index, axis=0)

PlayStoreData.head()

# Resetting the index

PlayStoreData.set\_index( np.arange(0,len(PlayStoreData)) , inplace=True)

PlayStoreData

# Reviews: Very few apps have very high number of reviews. These are all star apps that don’t help with the analysis and,

# in fact, will skew it. Drop records having more than 2 million reviews.

highreview=PlayStoreData[PlayStoreData['Reviews']>2000000]

highreview

# Now droping those records from the dataset

PlayStoreData=PlayStoreData.drop(highreview.index, axis=0)

PlayStoreData.head()

# Now we have all records which has less than 2 million reviews

#Installs: There seems to be some outliers in this field too. Apps having very high number of installs should be dropped from the analysis.

#Find out the different percentiles – 10, 25, 50, 70, 90, 95, 99

np.percentile(PlayStoreData['Installs'], [10, 25, 50, 70, 90, 95, 99])

# array([1.e+03, 1.e+04, 5.e+05, 1.e+06, 1.e+07, 1.e+07, 1.e+08])

# Make scatter plot/joinplot for Rating vs. Price

# What pattern do you observe? Does rating increase with price?

plt.figure(figsize=(10,6))

sns.set\_style(style='whitegrid',)

sns.set(font\_scale=1.2)

sns.scatterplot(x=PlayStoreData2['Price'], y=PlayStoreData2['Rating'],hue= PlayStoreData2['Rating'],palette='autumn\_r')

plt.title('Prices vs Ratings', fontsize=18)

plt.ylabel('Rating',fontsize=14)

plt.xlabel('Price', fontsize=14)

plt.show()

# Rating is not dependent on increase with price. Rating between 2.5 to 4, there are quite good amount of app lies with price

# around 10 to 35. Most of app rating lies within price 40.

# Apps with rating in the range of 4 to 5, price of those apps varies 0 to 40.

# Only one app with rating 4.5 has highest price say around 80.

# Make scatter plot/joinplot for Rating vs. Size

# Are heavier apps rated better?

plt.figure(figsize=(10,6))

sns.scatterplot(x=PlayStoreData2['Size'], y=PlayStoreData2['Rating'], hue= PlayStoreData2['Rating'],palette='rocket\_r')

plt.title('Sizes vs Ratings', fontsize=18)

plt.ylabel('Rating',fontsize=14)

plt.xlabel('Size', fontsize=14)

plt.show()

# Here, very few apps has rating 1 and size lies b/w 0 to 40000.

# Apps which ratings b/w 1 to 3 are scatterly spread with the size of maximum around 80000,

# But if we notice higher ratings apps, moderate number of apps size are around 100000 .

# Most of the apps has ratings between 3.5 to nearly 5 as seen in histograms.

# It is cleared that most of the app are rated higher.

# So, we can say that most of the heavier apps rated better.

# Make scatter plot/joinplot for Rating vs. Reviews

# Does more review mean a better rating always?

plt.figure(figsize=(10,6))

sns.scatterplot(x=PlayStoreData2['Reviews'], y=PlayStoreData2['Rating'], hue= PlayStoreData2['Rating'], palette='mako\_r')

plt.title('Reviews vs Ratings', fontsize=18)

plt.ylabel('Rating',fontsize=14)

plt.xlabel('Reviews', fontsize=14)

plt.show()

# In our plot,

# we noticed that lower ratings apps has very low reviews nearly zero.

# Ratings b/w 3.5 to 4 has moderate number of Reviews.

# Ratings from 4 and higher has higher number of reviews, and some of the apps has higher reviews .

# If we see the histograms for ratings, most of records lies b/w 3.5 to nearly 5.

# We conclude that more review means a better rating always.

# Make boxplot for Rating vs. Content Rating

# Is there any difference in the ratings? Are some types liked better?

plt.figure(figsize= (10,6))

sns.boxplot(x = 'Content Rating', y = 'Rating', data = PlayStoreData2, palette='hls')

plt.title('Content Rating vs Ratings', fontsize=18)

plt.xticks(fontsize=12, rotation='45')

plt.ylabel('Rating',fontsize=14)

plt.xlabel('Content Rating', fontsize=14)

plt.show()

# Everyone: here, max rating given is 5 and low rating is around 3.3, but lots of people gave lower rating in this kind of apps.

# median is lies around 4.4.

# Teen: In this type of apps, max and min ratings are 5 and closely 3.4, median lies around close to 4.

# Some people gave relatively low ratings as well.

# Everyone 10: This kind of apps, max and min ratings are around 3.5 to 5 and median is around 4.4.

# few people gave some low ratings also.

# Mature 17: This kind of app, max and min ratings are around 3.1 to 5 and median is around 4.3.

# it has also some low ratings.

# Adults only 18+: This kind of apps, max and min ratings are around 3.8 to 4.5 and median is around 4.5.

# Here, no lower rating has provided by users and this is most compact app and here no ouliers present.

# Unrated: all values are same.

# so, we noticed that there are quite variation between all kinds of apps.

# it can be said that "Adults only 18+" more better apps than others.

# Make boxplot for Ratings vs. Category

plt.figure(figsize= (18,6))

sns.boxplot(x = 'Category', y = 'Rating', data = PlayStoreData2)

plt.title('Category vs Ratings', fontsize=20)

plt.xticks(fontsize=12, rotation = 'vertical')

plt.yticks(fontsize=10)

plt.xlabel("Category",fontsize=16)

plt.ylabel("Rating",fontsize=16)

# 'ART\_AND\_DESIGN' and 'WEATHER' has no outliers.

# 'EDUCATION' has minimum no of ratings, means users are not interested to install this knds of apps

# FINANCE, BUSINESS, LIFESTYLE ,GAME, FAMILY have highst no of ratings means most of the people installed these kind of apps.

# In this figure, we noticed that in most of the apps users gave ratings b/w 1 to 5.

# Users are interrested more about FINANCE, BUSINESS, LIFESTYLE ,GAME, FAMILY apps, so we can say creater of these kinds of apps

# need to create more apps related to this.

# In DATING and LIFESTYLE apps difference b/w min value and Q1 are quite high.

# Here are somme apps which has lower ratings also.

# Which genre has the best ratings?

PlayStoreData2['Genres'].value\_counts()

# For the steps below, create a copy of the dataframe to make all the edits. Name it inp1.

inp1 = PlayStoreData2.copy(deep=True)

inp1

# we have copied our dataframe in inp1, because what we will change our dataframe for further steps

# Our main dataframe will be intact.

# Reviews and Install have some values that are still relatively very high. Before building a linear regression model,

# you need to reduce the skew. Apply log transformation (np.log1p) to Reviews and Installs.

inp1['Reviews'] = np.log(inp1['Reviews'])

inp1['Installs'] = np.log1p(inp1['Installs'])

# Drop columns App, Last Updated, Current Ver, and Android Ver. These variables are not useful for our task.

inp1.drop(['App','Last Updated','Current Ver','Android Ver'], axis = 1,inplace=True)

inp1.head()

# Converting Type column with dummy variables and concatenating with main DataFrame

inp1 = pd.concat( [ pd.get\_dummies(inp1['Type']) , inp1.iloc[:,[0,1,2,3,4,6,7,8]] ] , axis = 1)

inp1

# Get dummy columns for Category, Genres, and Content Rating. This needs to be done as the models do not understand categorical data,

# and all data should be numeric. Dummy encoding is one way to convert character fields to numeric.

# Name of dataframe should be inp2.

cat\_cols = ['Category', 'Genres', 'Content Rating']

inp2=pd.concat( [pd.get\_dummies(inp1,columns=cat\_cols,drop\_first=True),inp1.iloc[:,[0,1,3,4,5,6,7]] ] , axis = 1)

print(inp2.columns)

# 9. Train test split and apply 70-30 split. Name the new dataframes df\_train and df\_test.

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

# creating features and labels

features = inp2.iloc[:,1:].values

labels = inp2.iloc[:,[2]].values

# 10. Separate the dataframes into X\_train, y\_train, X\_test, and y\_test.

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test = train\_test\_split(features,labels,test\_size=0.3,random\_state=1)

# Name the new dataframes df\_train and df\_test.

df\_train=pd.DataFrame(X\_train)

df\_test=pd.DataFrame(X\_test)

# 11 . Model building

# Use linear regression as the technique

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

model.fit(X\_train,y\_train)

# Report the r2 on the train set

from sklearn.metrics import r2\_score

y\_train\_pred= model.predict(X\_train)

r2\_score(y\_train, y\_train\_pred)

# here r2 score based on training set is 1.0

# 12. Make predictions on test set, report R2

y\_test\_pred= model.predict(X\_test)

r2\_score(y\_test, y\_test\_pred)

# here r2 score based on testing set is 1.0