Project Objective

1. Load Python Libraries and Google Play store Data
2. Understanding the Dataframe and Variables
3. Data Cleaning and Preprocessing
4. Perform Descriptiive Stats
5. Perform EDA Analysis
6. Create a prediction Model using Random Forest and Multiple Linear Regression
7. Evalution of Model
8. Checking out of influential variables.

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn import metrics
import plotly.express as px
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
import statsmodels.api as sm
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
```

In [2]:

```
# Load training data
df = pd.read_csv('Googlestore.csv')
df.head()
```

Out[2]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0.3 and up
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	Varies with device	4.2 and up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 and up

In [3]:

```
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          9367 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
```

Checking Null values and filling it with preceding values

In [4]:

```
print(df.isnull().sum())
```

```
App                  0
Category             0
Rating            1474
Reviews              0
Size                 0
Installs             0
Type                 1
Price                0
Content Rating       1
Genres               0
Last Updated         0
Current Ver          8
Android Ver          3
dtype: int64
```

In [5]:

```
df = df.ffill(axis = 0)
print(df.isnull().sum())
```

```
App               0
Category          0
Rating            0
Reviews           0
Size              0
Installs          0
Type              0
Price             0
Content Rating    0
Genres            0
Last Updated      0
Current Ver       0
Android Ver       0
dtype: int64
```

In [6]:

```
print(df.columns.tolist())
```

```
['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type', 'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current
Ver', 'Android Ver']
```

In [7]:

```
df.head(2)
```

Out[7]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up

DATA CLEANING AND PRE_PROCESSING

Checking Uniformity and consistency

1. App name and few other columns dont hold much value to analysis. They can be dropped.
2. Category needs to be Unique and should be string value
3. Rating should not be greater than five or should not contain any string
4. Reviews , Price , Install and size should be Numeric type. Content_Rating ,Type and Genres should be Unique and string type.
5. Last three columns are not much of use and has been dropped.

In [8]:

```
df = df.drop(columns=[ 'App','Last Updated', 'Current Ver', 'Android Ver'])
df.head(2)
```

Out[8]:

	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres
0	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design
1	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play

In [9]:

```
df.dtypes
```

Out[9]:

```
Category           object
Rating            float64
Reviews            object
Size               object
Installs           object
Type               object
Price              object
Content Rating     object
Genres             object
dtype: object
```

In [10]:

```
print(df.columns.tolist())
```

```
['Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type', 'Price', 'Content Rating', 'Genres']
```

Rename Columns Properly

In [11]:

```
df = df.rename(columns={'Reviews': 'No_of_Reviews' , 'Size':'Size_Kb', 'Installs':'No_of_Installs','Content Rating':'Content_Type'
df.head(2)
```

Out[11]:

	Category	Rating	No_of_Reviews	Size_Kb	No_of_Installs	Type	Price	Content_Type	Genre
0	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design
1	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play

In [12]:

```
print(df.columns.tolist())
```

```
['Category', 'Rating', 'No_of_Reviews', 'Size_Kb', 'No_of_Installs', 'Type', 'Price', 'Content_Type', 'Genre']
```

In [13]:

```
print(df.Category.unique())
```

```
['ART_AND_DESIGN' 'AUTO_AND_VEHICLES' 'BEAUTY' 'BOOKS_AND_REFERENCE'
 'BUSINESS' 'COMICS' 'COMMUNICATION' 'DATING' 'EDUCATION' 'ENTERTAINMENT'
 'EVENTS' 'FINANCE' 'FOOD_AND_DRINK' 'HEALTH_AND_FITNESS' 'HOUSE_AND_HOME'
 'LIBRARIES_AND_DEMO' 'LIFESTYLE' 'GAME' 'FAMILY' 'MEDICAL' 'SOCIAL'
 'SHOPPING' 'PHOTOGRAPHY' 'SPORTS' 'TRAVEL_AND_LOCAL' 'TOOLS'
 'PERSONALIZATION' 'PRODUCTIVITY' 'PARENTING' 'WEATHER' 'VIDEO_PLAYERS'
 'NEWS_AND_MAGAZINES' 'MAPS_AND_NAVIGATION' '1.9']
```

Check Rating Values

In [14]:

```
df[df['Rating'] > 5]
```

Out[14]:

	Category	Rating	No_of_Reviews	Size_Kb	No_of_Installs	Type	Price	Content_Type	Genre
10472	1.9	19.0	3.0M	1,000+	Free	0	Everyone	Everyone	February 11, 2018

In [15]:

```
df[df['Rating'] < 1]
```

Out[15]:

Category	Rating	No_of_Reviews	Size_Kb	No_of_Installs	Type	Price	Content_Type	Genre

In [16]:

```
a = df.Rating.unique().tolist()
a.sort()
a[-9:]
```

Out[16]:

```
[4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5.0, 19.0]
```

In [17]:

```
a[1:4]
```

Out[17]:

```
[1.2, 1.4, 1.5]
```

In [18]:

```
df = df.drop(df[df.Rating > 5].index)
```

In [19]:

```
df[df.Rating > 5]
```

Out[19]:

Category	Rating	No_of_Reviews	Size_Kb	No_of_Installs	Type	Price	Content_Type	Genre

In [20]:

```
print(df.Category.unique())
```

```
['ART_AND_DESIGN' 'AUTO_AND_VEHICLES' 'BEAUTY' 'BOOKS_AND_REFERENCE'
 'BUSINESS' 'COMICS' 'COMMUNICATION' 'DATING' 'EDUCATION' 'ENTERTAINMENT'
 'EVENTS' 'FINANCE' 'FOOD_AND_DRINK' 'HEALTH_AND_FITNESS' 'HOUSE_AND_HOME'
 'LIBRARIES_AND_DEMO' 'LIFESTYLE' 'GAME' 'FAMILY' 'MEDICAL' 'SOCIAL'
 'SHOPPING' 'PHOTOGRAPHY' 'SPORTS' 'TRAVEL_AND_LOCAL' 'TOOLS'
 'PERSONALIZATION' 'PRODUCTIVITY' 'PARENTING' 'WEATHER' 'VIDEO_PLAYERS'
 'NEWS_AND_MAGAZINES' 'MAPS_AND_NAVIGATION']
```

In [21]:

```
df['Category'] = df['Category'].astype('string')
```

In [22]:

```
df.dtypes
```

Out[22]:

```
Category           string
Rating            float64
No_of_Reviews      object
Size_Kb            object
No_of_Installs     object
Type               object
Price              object
Content_Type       object
Genre              object
dtype: object
```

Category and Rating looks fine

In [23]:

```
(df['No_of_Reviews'].eq('exact_string')).any()
```

Out[23]:

```
False
```

In [24]:

```
(df['No_of_Reviews'].eq(0)).any()
```

Out[24]:

```
False
```

In [25]:

```
df['No_of_Reviews'] = df['No_of_Reviews'].astype(int)
```

In [26]:

```
#df.dtypes
```

In [27]:

```
df.No_of_Reviews.mean()
```

Out[27]:

```
444152.89603321033
```

In [28]:

```
#print(df.Size_Kb.unique())
```

Cleaning the SIZE Column

In [29]:

```
df['Size_Kb'] = df['Size_Kb'].replace({'Varies with device':0})
```

In [30]:

```
df['Size_Kb'] = df['Size_Kb'].astype('string')
```

In [31]:

```
df.Size_Kb.dtypes
```

Out[31]:

```
string[python]
```

In [32]:

```
units = {'M': 'e+06',             # convert M to 'e+06' (equivalent to '* 1000000')
         'k': 'e+03',             # convert K to 'e+03' (equivalent to '* 1000')
         '\s': ''                 # remove white space, if any
          }
```

In [33]:

```
df['Size_Kb'] = df['Size_Kb'].replace(units, regex=True)
df.head(3)
```

Out[33]:

	Category	Rating	No_of_Reviews	Size_Kb	No_of_Installs	Type	Price	Content_Type	Genre
0	ART_AND_DESIGN	4.1	159	19e+06	10,000+	Free	0	Everyone	Art & Design
1	ART_AND_DESIGN	3.9	967	14e+06	500,000+	Free	0	Everyone	Art & Design;Pretend Play

	Category	Rating	No_of_Reviews	Size_Kb	No_of_Installs	Type	Price	Content_Type	Genre
2	ART_AND_DESIGN	4.7	87510	8.7e+06	5,000,000+	Free	0	Everyone	Art & Design

In [34]:

```
df.dtypes
```

Out[34]:

```
Category           string
Rating            float64
No_of_Reviews       int32
Size_Kb            string
No_of_Installs     object
Type               object
Price              object
Content_Type       object
Genre              object
dtype: object
```

In [35]:

```
pd.options.display.float_format = '{:.2f}'.format
```

In [36]:

```
df['Size_Kb'] = df['Size_Kb'].fillna(0).astype(float)
```

In [37]:

```
df.head(3)
```

Out[37]:

	Category	Rating	No_of_Reviews	Size_Kb	No_of_Installs	Type	Price	Content_Type	Genre
0	ART_AND_DESIGN	4.10	159	19000000.00	10,000+	Free	0	Everyone	Art & Design
1	ART_AND_DESIGN	3.90	967	14000000.00	500,000+	Free	0	Everyone	Art & Design;Pretend Play
2	ART_AND_DESIGN	4.70	87510	8700000.00	5,000,000+	Free	0	Everyone	Art & Design

In [38]:

```
pd.set_option('display.float_format','{:.0f}'.format)  ## It does not touch zero values.So we check zeroes again.
```

In [39]:

```
(df['Size_Kb'].eq(0)).any()
```

Out[39]:

```
True
```

In [40]:

```
df['Size_Kb'] = df['Size_Kb'].replace(to_replace=0, method='ffill')
```

In [41]:

```
(df['Size_Kb'].eq(0)).any() # .eq()function to check if any value is equal to zero or df['Size_Kb'].eq(0)
```

Out[41]:

```
False
```

In [42]:

```
#df.head(2)
```

Cleaning No. of Installs Column

In [43]:

```
df['No_of_Installs'] = df['No_of_Installs'].apply(lambda x : x.strip('+').replace(',', ''))
```

In [44]:

```
#df.head(2)
```

In [45]:

```
#df.dtypes
```

In [46]:

```
df['No_of_Installs'] = df['No_of_Installs'].astype(float)
```

In [47]:

```
print(df.Type.unique())
```

```
['Free' 'Paid']
```

In [48]:

```
#print(df.Price.unique())  ##PRICE CONTAINS DOLLAR SIGNS
```

In [49]:

```
df['Price'] = df['Price'].apply(lambda x : x.strip('$'))
```

In [50]:

```
df['Price'] = df['Price'].astype(float)
```

In [51]:

```
print(df.Content_Type.unique())
```

```
['Everyone' 'Teen' 'Everyone 10+' 'Mature 17+' 'Adults only 18+' 'Unrated']
```

In [52]:

```
#print(df.Genre.unique())
```

In [53]:

```
df.dtypes
```

Out[53]:

```
Category            string
Rating             float64
No_of_Reviews        int32
Size_Kb            float64
No_of_Installs     float64
Type                object
Price              float64
Content_Type        object
Genre               object
dtype: object
```

In [54]:

```
df.head(4)
```

Out[54]:

	Category	Rating	No_of_Reviews	Size_Kb	No_of_Installs	Type	Price	Content_Type	Genre
0	ART_AND_DESIGN	4	159	19000000	10000	Free	0	Everyone	Art & Design
1	ART_AND_DESIGN	4	967	14000000	500000	Free	0	Everyone	Art & Design;Pretend Play
2	ART_AND_DESIGN	5	87510	8700000	5000000	Free	0	Everyone	Art & Design
3	ART_AND_DESIGN	4	215644	25000000	50000000	Free	0	Teen	Art & Design

ALL Columns looks clean - Data is ready for Analysis

In [55]:

```
df['Category'].value_counts()
```

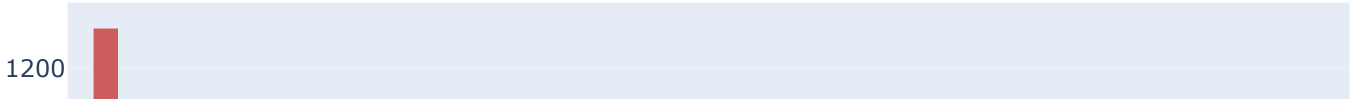
Out[55]:

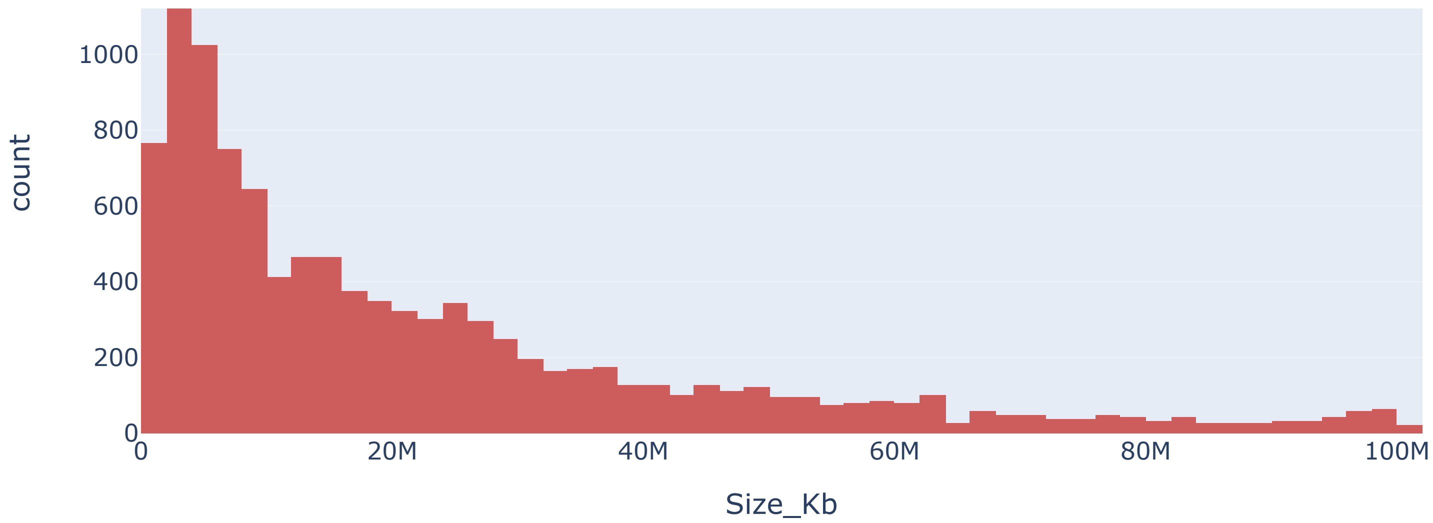
```
FAMILY                 1972
GAME                   1144
```

```
TOOLS                  843
MEDICAL                463
BUSINESS               460
PRODUCTIVITY           424
PERSONALIZATION        392
COMMUNICATION          387
SPORTS                 384
LIFESTYLE              382
FINANCE                366
HEALTH_AND_FITNESS     341
PHOTOGRAPHY            335
SOCIAL                 295
NEWS_AND_MAGAZINES     283
SHOPPING               260
TRAVEL_AND_LOCAL       258
DATING                 234
BOOKS_AND_REFERENCE    231
VIDEO_PLAYERS          175
EDUCATION              156
ENTERTAINMENT          149
MAPS_AND_NAVIGATION    137
FOOD_AND_DRINK         127
HOUSE_AND_HOME          88
AUTO_AND_VEHICLES       85
LIBRARIES_AND_DEMO      85
WEATHER                 82
ART_AND_DESIGN          65
EVENTS                  64
PARENTING               60
COMICS                  60
BEAUTY                  53
Name: Category, dtype: Int64
```

In [56]:

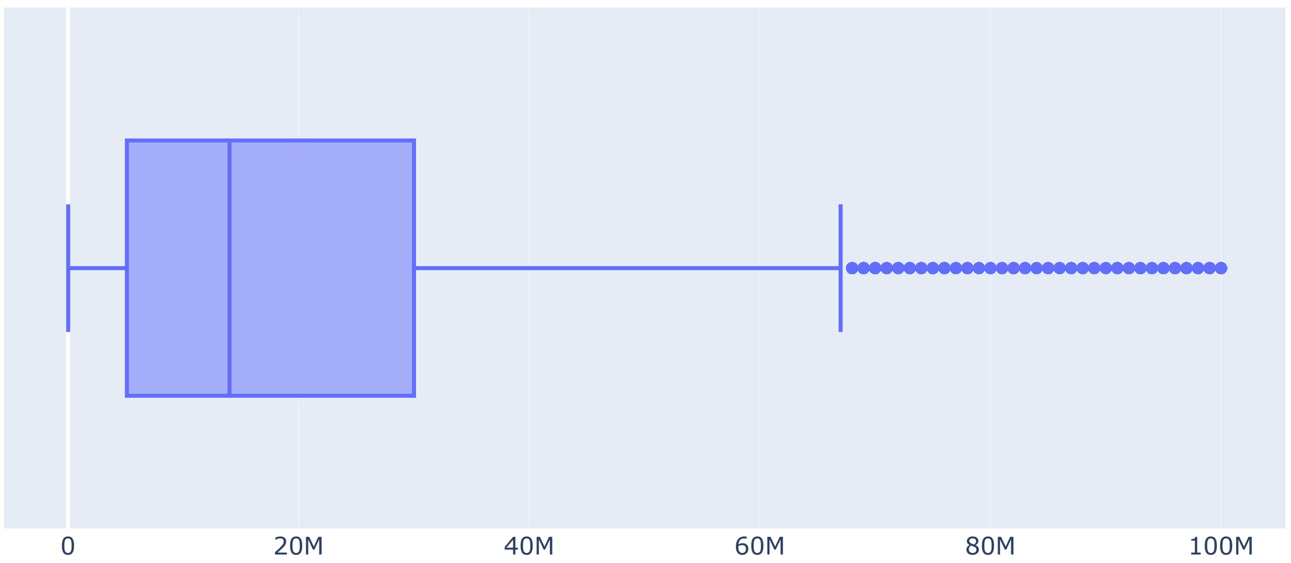
```
fig = px.histogram(df, x="Size_Kb" , width=800, height=400 , color_discrete_sequence=['indianred'])
fig.show()
```





In [57]:

```
fig = px.box(df, x="Size_Kb" , width=800, height=400)
fig.show()
```



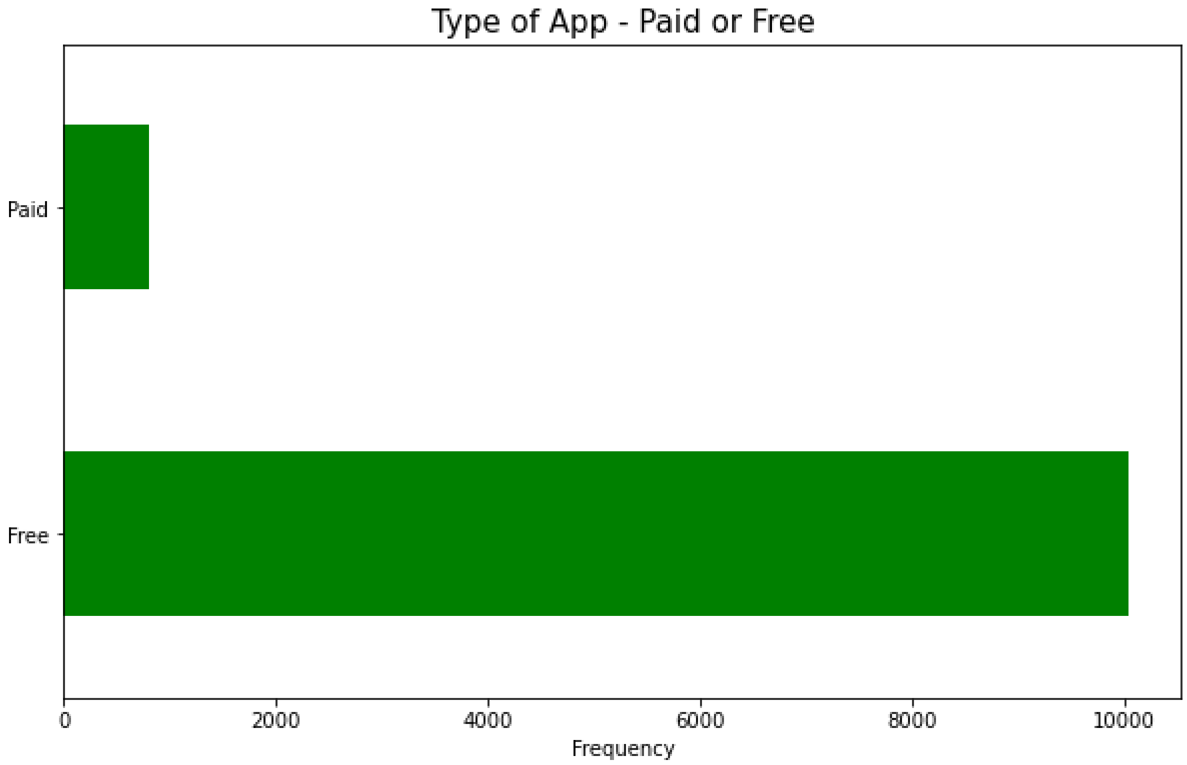
Size_Kb

In [58]:

```
plt.figure(figsize=(10,6))
df.Type.value_counts().plot(kind='barh',color = "green")
plt.title('Type of App - Paid or Free' , fontsize = 15)
plt.xlabel('Frequency')
```

Out[58]:

```
Text(0.5, 0, 'Frequency')
```



In [59]:

```
pd.crosstab(df.Category,df.Type)
```

Out[59]:

Type	Free	Paid
Category		
ART_AND_DESIGN	62	3
AUTO_AND_VEHICLES	82	3
BEAUTY	53	0
BOOKS_AND_REFERENCE	203	28
BUSINESS	446	14
COMICS	60	0
COMMUNICATION	360	27
DATING	227	7
EDUCATION	152	4
ENTERTAINMENT	147	2
EVENTS	63	1
FAMILY	1781	191
FINANCE	349	17
FOOD_AND_DRINK	125	2
GAME	1061	83
HEALTH_AND_FITNESS	325	16
HOUSE_AND_HOME	88	0
LIBRARIES_AND_DEMO	84	1
LIFESTYLE	363	19
MAPS_AND_NAVIGATION	132	5
MEDICAL	354	109
NEWS_AND_MAGAZINES	281	2
PARENTING	58	2

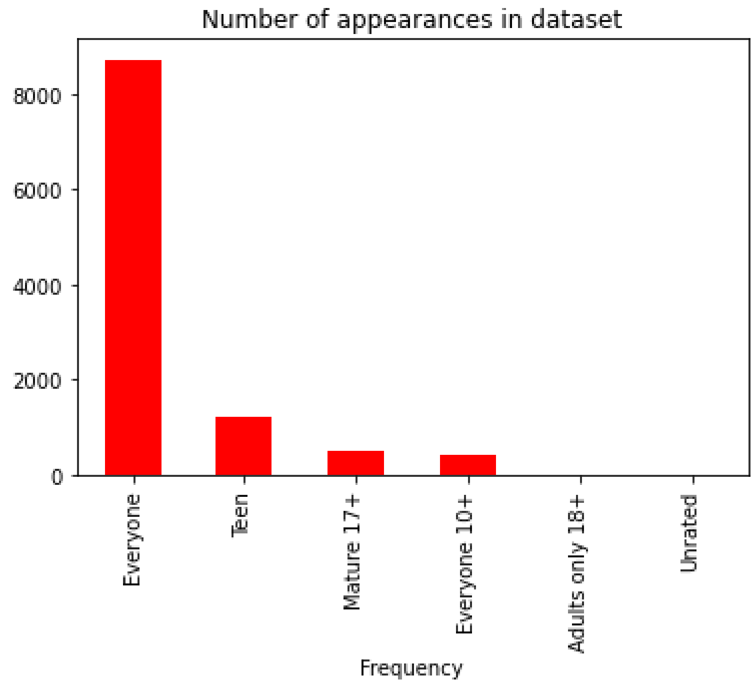
Type	Free	Paid
Category		
PERSONALIZATION	309	83
PHOTOGRAPHY	313	22
PRODUCTIVITY	396	28
SHOPPING	258	2
SOCIAL	292	3
SPORTS	360	24
TOOLS	765	78
TRAVEL_AND_LOCAL	246	12
VIDEO_PLAYERS	171	4
WEATHER	74	8

In [60]:

```
df.Content_Type.value_counts().plot(kind='bar',color = 'red')
plt.title('Number of appearances in dataset')
plt.xlabel('Frequency')
```

Out[60]:

```
Text(0.5, 0, 'Frequency')
```

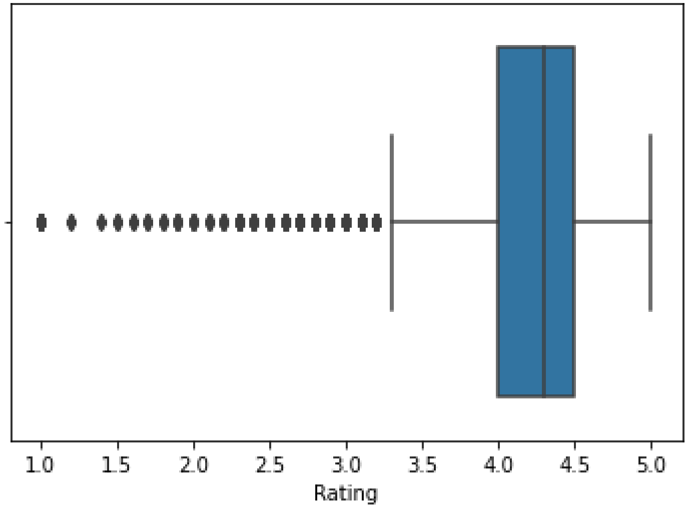


In [61]:

```
sns.boxplot(x=df["Rating"])
```

Out[61]:

```
<AxesSubplot:xlabel='Rating'>
```



In [62]:

```
p = df['Content_Type'].value_counts()
```

In [63]:

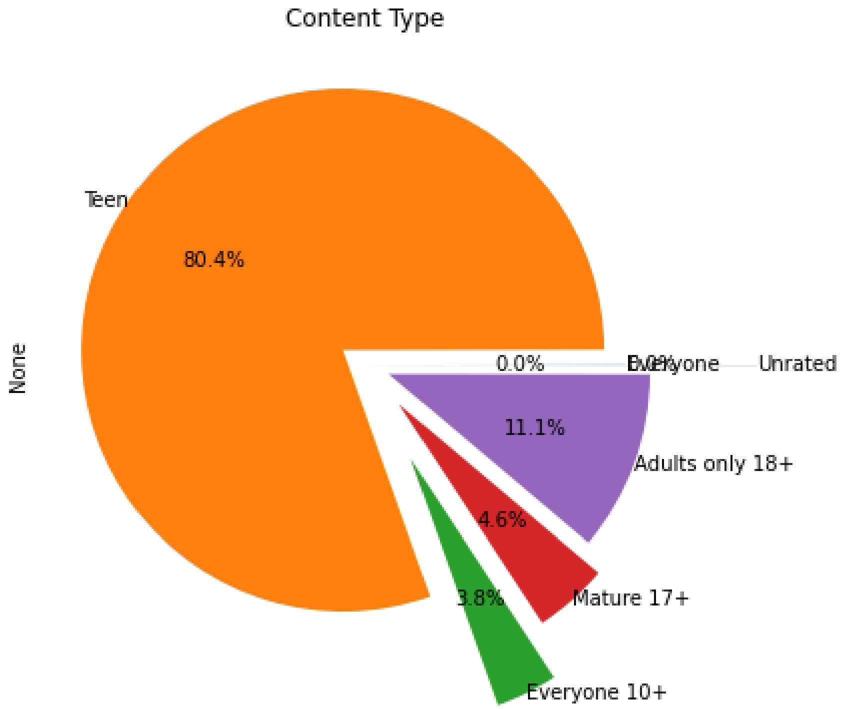
```
mylabels = df.Content_Type.unique()
```

In [64]:

```
plt.figure(figsize=(10,6))
myexplode = (0.0, 0.1, 0.4, 0.2,0.1,0.5)
e = df.groupby('Content_Type').size().plot(kind='pie', labels = mylabels , labeldistance=1 , explode = myexplode , autopct='%1.1f%
e.set_title('Content Type')
```

Out[64]:

```
Text(0.5, 1.0, 'Content Type')
```



In [65]:

```
df.dtypes
```

Out[65]:

```
Category           string
Rating            float64
No_of_Reviews       int32
```

```
Size_Kb            float64
No_of_Installs     float64
Type                object
Price              float64
Content_Type        object
Genre               object
dtype: object
```

In [66]:

```
df.head(3)
```

Out[66]:

	Category	Rating	No_of_Reviews	Size_Kb	No_of_Installs	Type	Price	Content_Type	Genre
0	ART_AND_DESIGN	4	159	19000000	10000	Free	0	Everyone	Art & Design
1	ART_AND_DESIGN	4	967	14000000	500000	Free	0	Everyone	Art & Design;Pretend Play
2	ART_AND_DESIGN	5	87510	8700000	5000000	Free	0	Everyone	Art & Design

Building a Prediction Model

In [67]:

```
df1 = df.copy()
```

ENCODE CATEGORICAL VARIABLES

DONT STANDARDISE TARGET VARIABLE BUT IT CAN BE ENCODED

DONT STANDARDISE CATEGORICAL VARIABLES , LABEL ENCODING WILL HANDLE THEM IN MODEL

ALWAYS STANDARDISE NUMERICAL COLUMNS WITH ORGANIC VALUES

In [68]:

```
le = LabelEncoder()
cols = ['Category','Type','Content_Type','Genre']
df1[cols] = df1[cols].apply(LabelEncoder().fit_transform)
df1.head()
```

Out[68]:

	Category	Rating	No_of_Reviews	Size_Kb	No_of_Installs	Type	Price	Content_Type	Genre
0	0	4	159	19000000	10000	0	0	1	9
1	0	4	967	14000000	500000	0	0	1	12
2	0	5	87510	8700000	5000000	0	0	1	9
3	0	4	215644	25000000	50000000	0	0	4	9
4	0	4	967	2800000	100000	0	0	1	11

In [69]:

```
nums = ['No_of_Reviews','Size_Kb','No_of_Installs','Price']
```

In [70]:

```
sc = StandardScaler()
df1[nums] = sc.fit_transform(df1[nums])
df1.head()
```

Out[70]:

	Category	Rating	No_of_Reviews	Size_Kb	No_of_Installs	Type	Price	Content_Type	Genre
0	0	4	-0	-0	-0	0	-0	1	9
1	0	4	-0	-0	-0	0	-0	1	12
2	0	5	-0	-1	-0	0	-0	1	9
3	0	4	-0	0	0	0	-0	4	9
4	0	4	-0	-1	-0	0	-0	1	11

In [71]:

```
df1.dtypes
```

Out[71]:

```
Category            int32
Rating            float64
No_of_Reviews     float64
Size_Kb           float64
No_of_Installs    float64
Type                int32
Price             float64
Content_Type        int32
```

```
Genre                   int32
dtype: object
```

In [72]:

```
pd.set_option('display.float_format','{:.6f}'.format) ## Display upto 6 decimal places
```

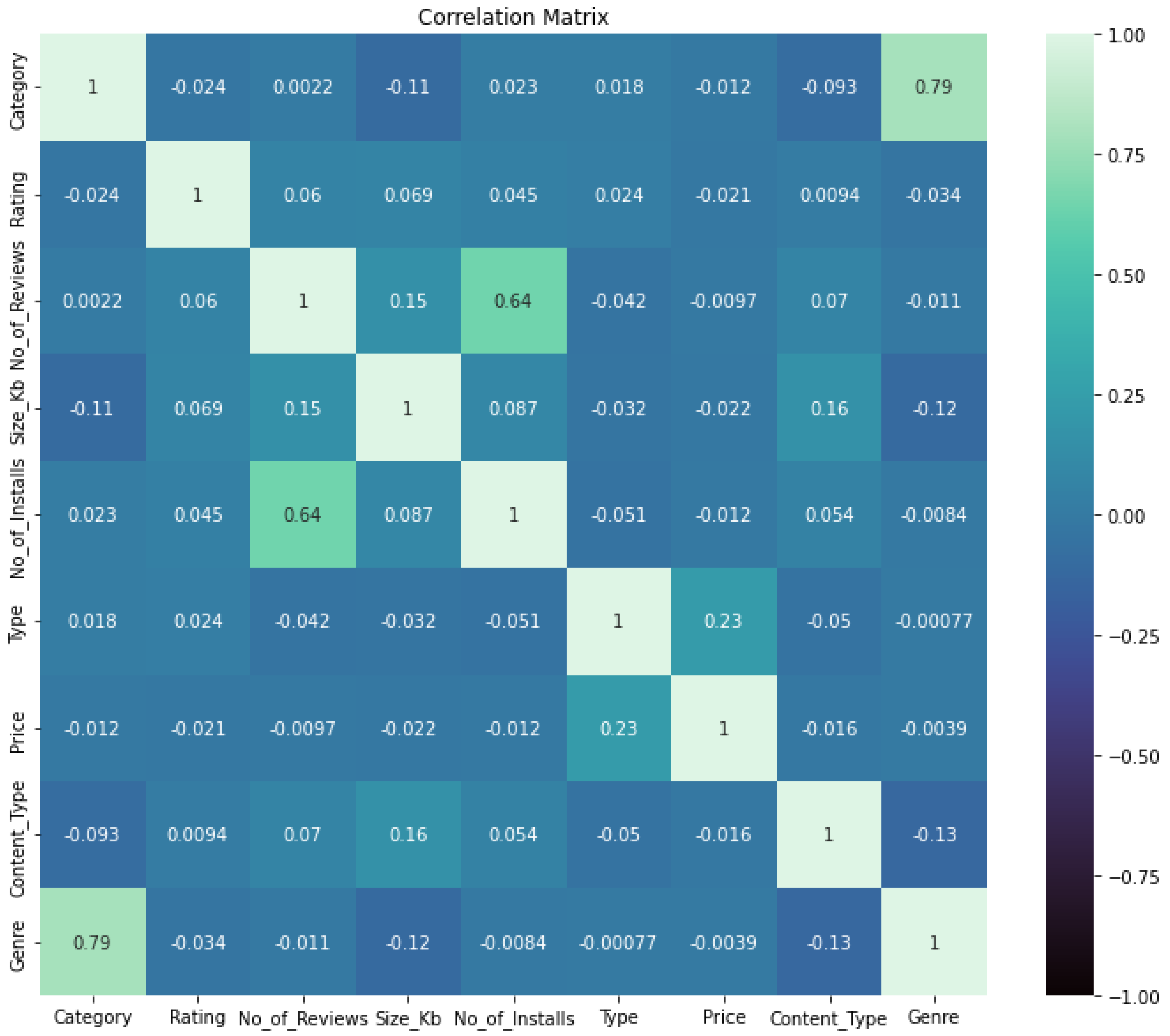
In [73]:

```
#df1.head()
```

CORRELATION MATRIX

In [74]:

```
plt.figure(figsize=(12, 10))
sns.heatmap(df1.corr(), annot=True, vmin=-1.0, cmap='mako')
plt.title("Correlation Matrix")
plt.show()
```



In [75]:

```
X = df1.drop('Rating', axis=1)
y = df1['Rating']
```

In [76]:

```
X.head(2)
```

Out[76]:

	Category	No_of_Reviews	Size_Kb	No_of_Installs	Type	Price	Content_Type	Genre
0	0	-0.151657	-0.126374	-0.181761	0	-0.064416	1	9
1	0	-0.151381	-0.348377	-0.175998	0	-0.064416	1	12

In [77]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [78]:

```
print("Training Set Dimensions:", X_train.shape)
print("Validation Set Dimensions:", X_test.shape)
```

```
Training Set Dimensions: (7588, 8)
Validation Set Dimensions: (3252, 8)
```

Multiple Linear regression Model fitting

In [79]:

```
model = LinearRegression()
model.fit(X_train, y_train)
y_predict = model.predict(X_test)
meanAbErr = metrics.mean_absolute_error(y_test, y_predict)
meanSqErr = metrics.mean_squared_error(y_test, y_predict)
rootMeanSqErr = np.sqrt(metrics.mean_squared_error(y_test, y_predict))
print('Mean Absolute Error:', meanAbErr)
print('Mean Square Error of Multiple Regression:', meanSqErr)
print('Root Mean Square Error of Multiple Regression:', rootMeanSqErr)
```

```
Mean Absolute Error: 0.3816784389831434
Mean Square Error of Multiple Regression: 0.30038055126787044
Root Mean Square Error of Multiple Regression: 0.5480698415967352
```

Random Forest Model

In [80]:

```
randomf = RandomForestRegressor(n_estimators=300)
randomf.fit(X_train, y_train)
y_predicted = randomf.predict(X_test)
mser = mean_squared_error(y_test,y_predicted)
rmser = mser**.5
```

```
print('Mean squared Error of Random Forest :', mser)
print('Root Mean squared Error Random Forest:', rmser)
```

```
Mean squared Error of Random Forest : 0.2826240324344135
Root Mean squared Error Random Forest: 0.5316239577317914
```

In [81]:

```
pd.set_option('display.float_format','{:.3f}'.format)
```

In [82]:

```
randomf.feature_importances_
```

Out[82]:

```
array([0.10542664, 0.3124992 , 0.30245635, 0.09744145, 0.00691173,
       0.02813163, 0.03111427, 0.11601874])
```

In [83]:

```
importances = randomf.feature_importances_
features = X.columns
```

In [84]:

```
Feature_Importance = pd.Series(importances, index=features)
Feature_Importance
```

Out[84]:

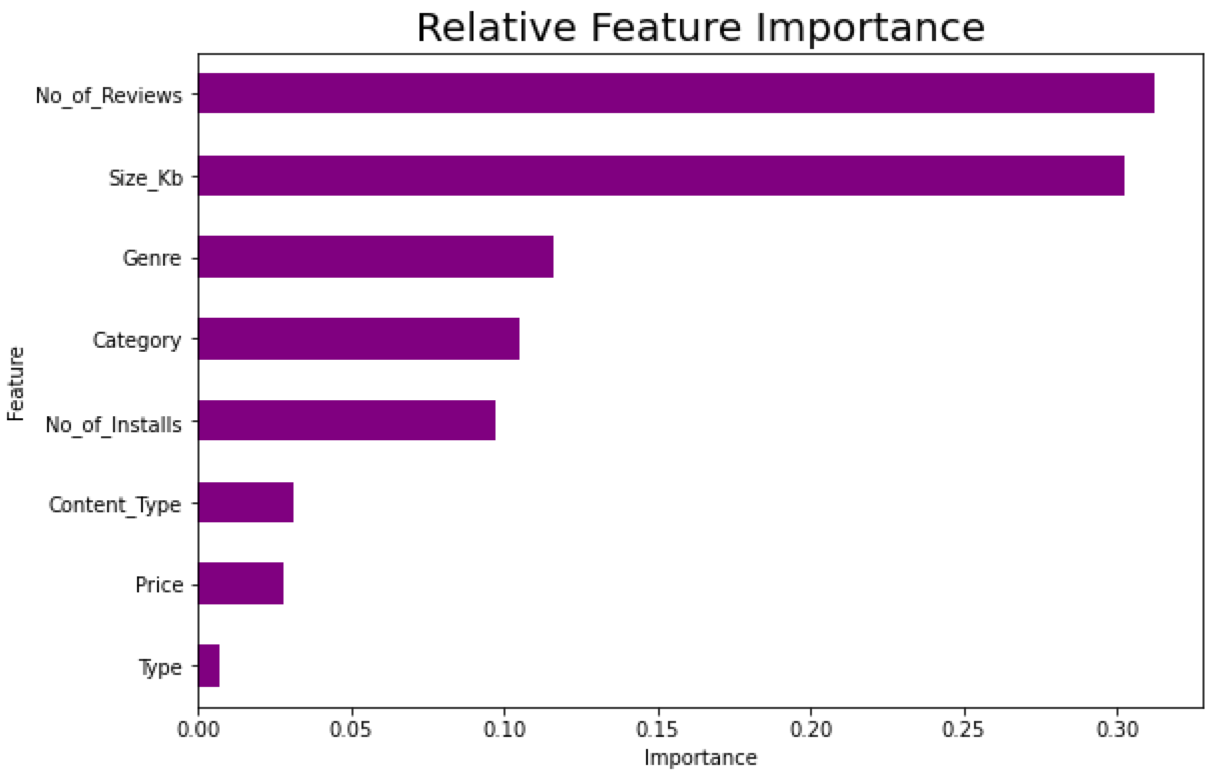
```
Category          0.105
No_of_Reviews     0.312
Size_Kb           0.302
No_of_Installs    0.097
Type              0.007
Price             0.028
Content_Type      0.031
Genre             0.116
dtype: float64
```

In [85]:

```
plt.figure(figsize=(9, 6))
Feature_Importance.sort_values(ascending=True, inplace=True)
Feature_Importance.plot.barh(color='purple')
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.title("Relative Feature Importance" , fontsize=20)
```

Out[85]:

```
Text(0.5, 1.0, 'Relative Feature Importance')
```



Size_Kb , Reviews , Genre and Category of an app are influential variables.

In [86]:

```
pred_df=pd.DataFrame({'Actual Rating':y_test,'Predicted Rating':y_predicted})
pd.set_option('display.float_format','{:.1f}'.format)
pred_df.head(8)
```

Out[86]:

	Actual Rating	Predicted Rating
10032	4.0	4.2
5649	4.8	4.7
10643	4.3	3.5
8531	4.3	4.3
10728	4.2	4.3

	Actual Rating	Predicted Rating
157	4.2	4.1
61	4.9	4.7
5136	3.4	4.0

RMSE of Random Forest Model is better than Multiple Linear Regression

Overall , its a good model with RMSE close to 0.51 and MAE close to 0.28