App Rating Prediction

Project 1

DESCRIPTION

Objective: Make a model to predict the app rating, with other information about the app provided.

Problem Statement:

Google Play Store team is about to launch a new feature wherein, certain apps that are promising, are boosted in visibility. The boost will manifest in multiple ways including higher priority in recommendations sections (“Similar apps”, “You might also like”, “New and updated games”). These will also get a boost in search results visibility. This feature will help bring more attention to newer apps that have the potential.

Domain: General

Analysis to be done: The problem is to identify the apps that are going to be good for Google to promote. App ratings, which are provided by the customers, is always a great indicator of the goodness of the app. The problem reduces to: predict which apps will have high ratings.

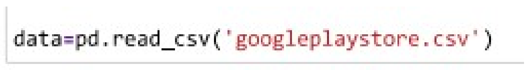
Content: Dataset: Google Play Store data (“googleplaystore.csv”)

Fields in the data –

- App: Application name
- Category: Category to which the app belongs
- Rating: Overall user rating of the app
- Reviews: Number of user reviews for the app
- Size: Size of the app
- Installs: Number of user downloads/installs for the app
- Type: Paid or Free
- Price: Price of the app
- Content Rating: Age group the app is targeted at - Children / Mature 21+ / Adult
- Genres: An app can belong to multiple genres (apart from its main category). For example, a musical family game will belong to Music, Game, Family genres.
- Last Updated: Date when the app was last updated on Play Store
- Current Ver: Current version of the app available on Play Store
- Android Ver: Minimum required Android version

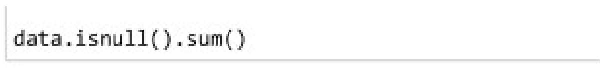
Steps to perform:

1. Load the data file using pandas.



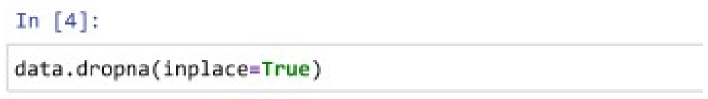
```
data=pd.read_csv('googleplaystore.csv')
```

2. Check for null values in the data. Get the number of null values for each column.



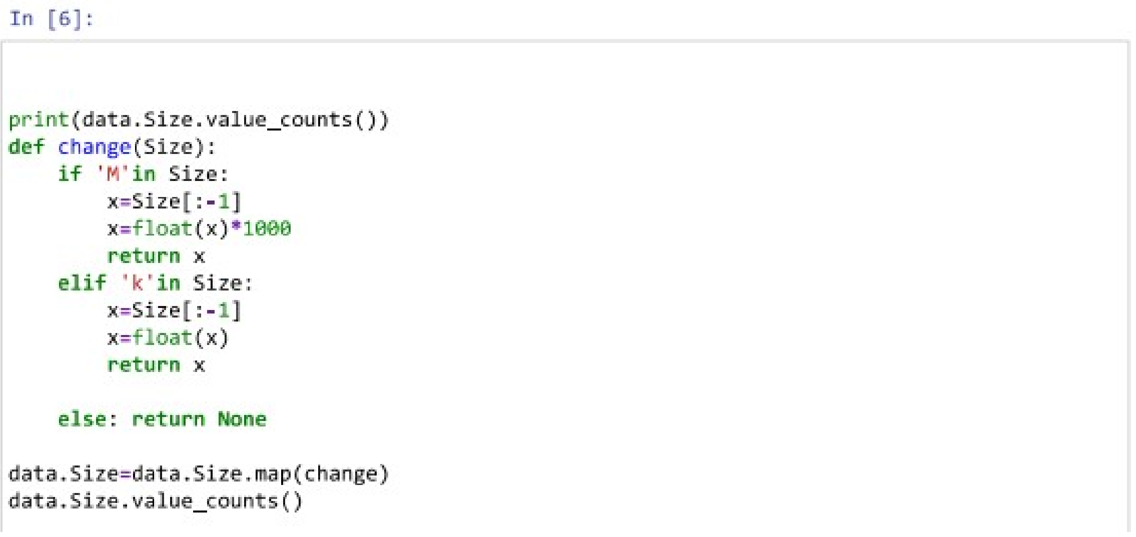
```
data.isnull().sum()
```

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



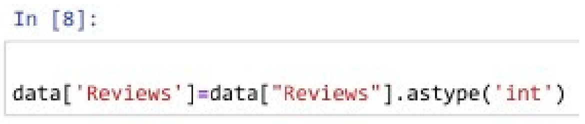
```
In [4]:
data.dropna(inplace=True)
```

4. Variables seem to have incorrect type and inconsistent formatting. You need to fix them:
    1. Size column has sizes in Kb as well as Mb. To analyze, you'll need to convert these to numeric.
        0. Extract the numeric value from the column
        1. Multiply the value by 1,000, if size is mentioned in Mb



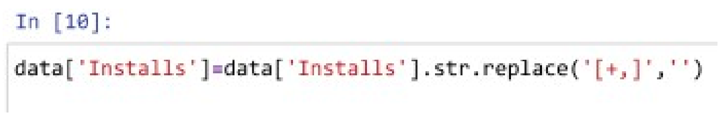
```
In [6]:


print(data.Size.value_counts())
def change(Size):
    if 'M'in Size:
        x=Size[:-1]
        x=float(x)*1000
        return x
    elif 'k'in Size:
        x=Size[:-1]
        x=float(x)
        return x

    else: return None

data.Size=data.Size.map(change)
data.Size.value_counts()
```

    2. Reviews is a numeric field that is loaded as a string field. Convert it to numeric (int/float).

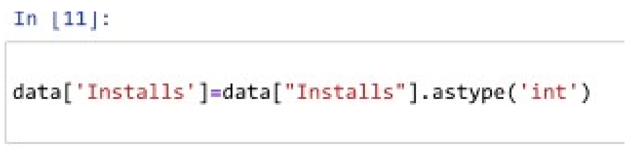


```
In [8]:

data['Reviews']=data["Reviews"].astype('int')
```

3. Installs field is currently stored as string and has values like 1,000,000+.
    0. Treat 1,000,000+ as 1,000,000
    1. remove ‘+’, ‘,’ from the field, convert it to integer

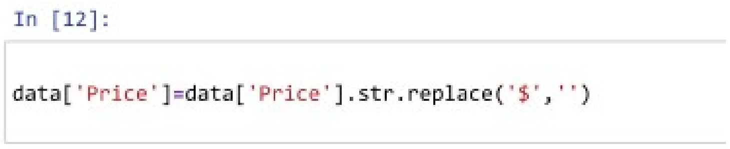


```
In [10]:
data['Installs']=data['Installs'].str.replace('[+,]','')
```



```
In [11]:
data['Installs']=data["Installs"].astype('int')
```

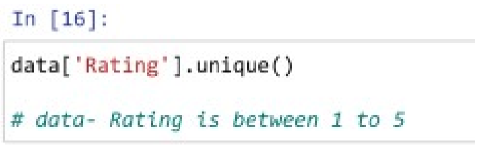
4. Price field is a string and has $ symbol. Remove ‘$’ sign, and convert it to numeric.



```
In [12]:
data['Price']=data['Price'].str.replace('$','')
```

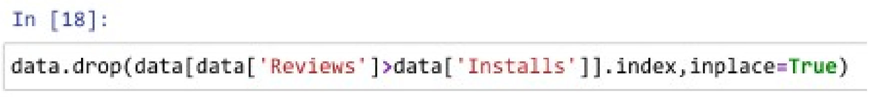
5. Sanity checks:

    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 this range.



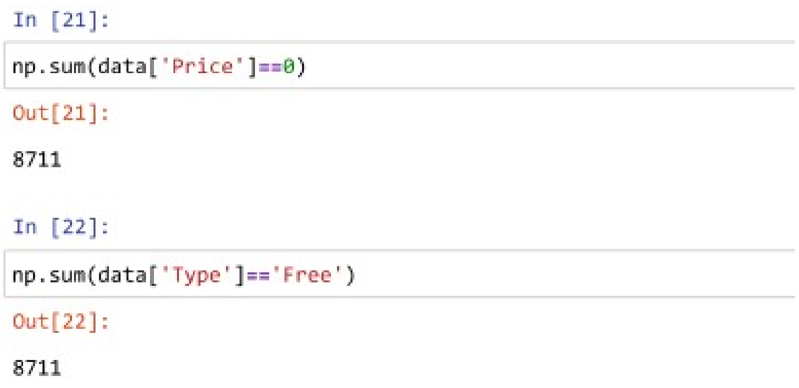
```
In [16]:
data['Rating'].unique()

# data- Rating is between 1 to 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.



```
In [18]:
data.drop(data[data['Reviews']>data['Installs']].index,inplace=True)
```

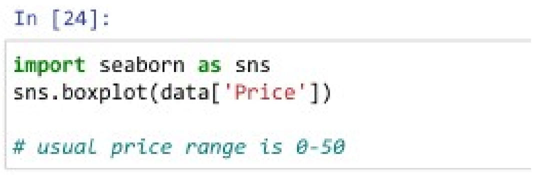
    3. For free apps (type = “Free”), the price should not be >0. Drop any such rows.



```
In [21]:
np.sum(data['Price']==0)
Out[21]:
8711

In [22]:
np.sum(data['Type']=='Free')
Out[22]:
8711
```

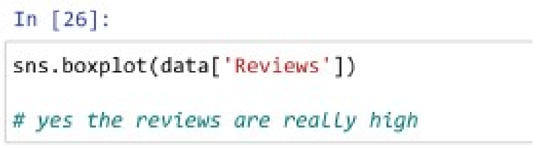
6. Performing univariate analysis:

- Boxplot for Price
    - Are there any outliers? Think about the price of usual apps on Play Store.



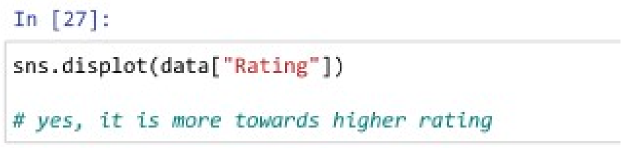
```
In [24]:
import seaborn as sns
sns.boxplot(data['Price'])

# usual price range is 0-50
```

- Boxplot for Reviews
    - Are there any apps with very high number of reviews? Do the values seem right?



```
In [26]:
sns.boxplot(data['Reviews'])

# yes the reviews are really high
```

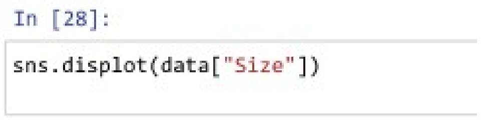
- Histogram for Rating
    - How are the ratings distributed? Is it more toward higher ratings?



```
In [27]:
sns.displot(data["Rating"])

# yes, it is more towards higher rating
```

- Histogram for Size

Note down your observations for the plots made above. Which of these seem to have outliers?



```
In [28]:
sns.displot(data["Size"])
```

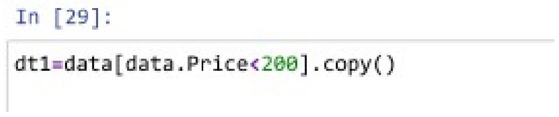
7. Outlier treatment:

1. Price: From the box plot, it seems like there are some apps with very high price. A price of $200 for an application on the Play Store is very high and suspicious!

    0. Check out the records with very high price

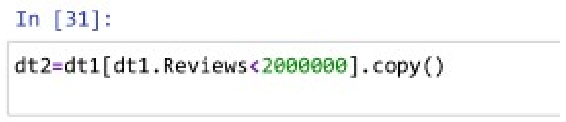
        0. Is 200 indeed a high price?

    1. Drop these as most seem to be junk apps



```
In [29]:
dt1=data[data.Price<200].copy()
```

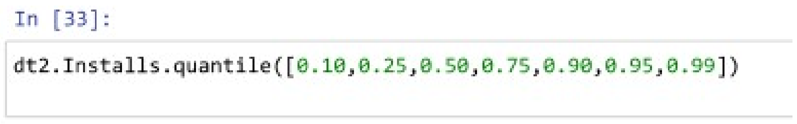
2. Reviews: Very few apps have very high number of reviews. These are all star apps that don't help with the analysis and, in fact, will skew it. Drop records having more than 2 million reviews.



```
In [31]:
dt2=dt1[dt1.Reviews<2000000].copy()
```

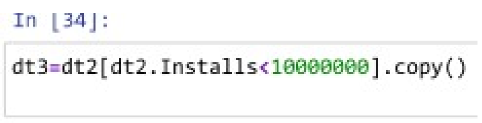
3. Installs: There seems to be some outliers in this field too. Apps having very high number of installs should be dropped from the analysis.

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



```
In [33]:
dt2.Installs.quantile([0.10,0.25,0.50,0.75,0.90,0.95,0.99])
```

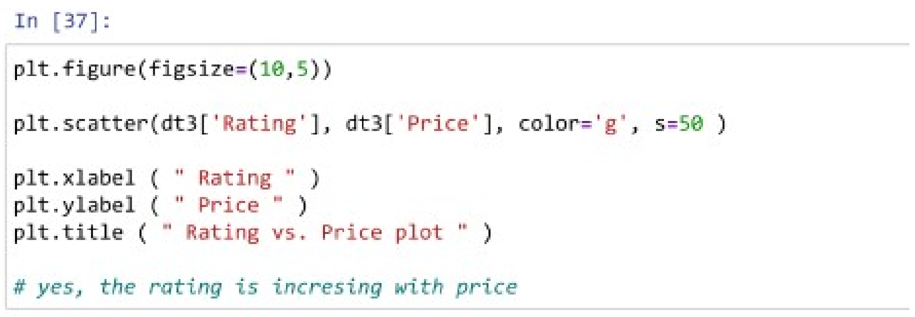
    1. Decide a threshold as cutoff for outlier and drop records having values more than that



```
In [34]:
dt3=dt2[dt2.Installs<10000000].copy()
```

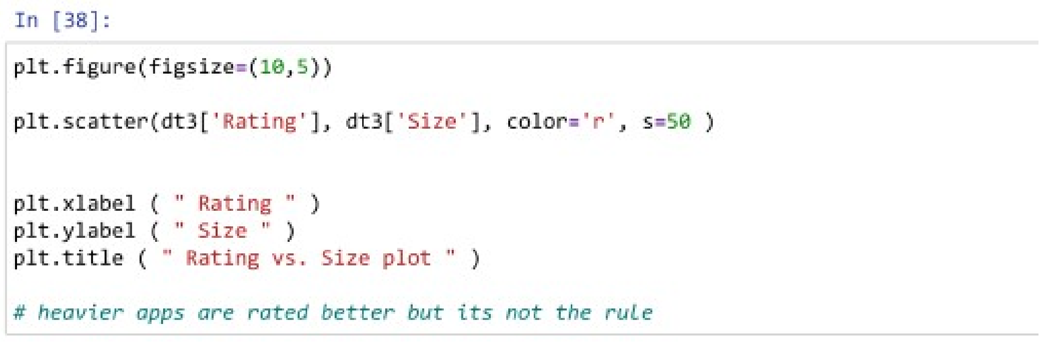
8. Bivariate analysis: Let's look at how the available predictors relate to the variable of interest, i.e., our target variable rating. Make scatter plots (for numeric features) and box plots (for character features) to assess the relations between rating and the other features.

1. Make scatter plot/joinplot for Rating vs. Price
    0. What pattern do you observe? Does rating increase with price?



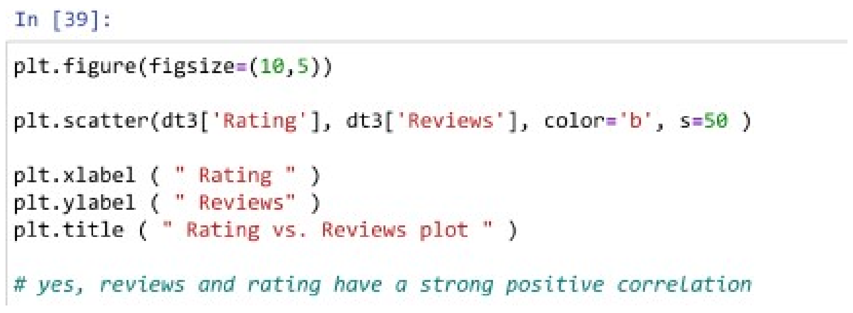
```
In [37]:
plt.figure(figsize=(10,5))

plt.scatter(dt3['Rating'], dt3['Price'], color='g', s=50 )

plt.xlabel ( " Rating " )
plt.ylabel ( " Price " )
plt.title ( " Rating vs. Price plot " )

# yes, the rating is incresing with price
```

2. Make scatter plot/joinplot for Rating vs. Size
    0. Are heavier apps rated better?



```
In [38]:
plt.figure(figsize=(10,5))

plt.scatter(dt3['Rating'], dt3['Size'], color='r', s=50 )


plt.xlabel ( " Rating " )
plt.ylabel ( " Size " )
plt.title ( " Rating vs. Size plot " )

# heavier apps are rated better but its not the rule
```

3. Make scatter plot/joinplot for Rating vs. Reviews
    0. Does more review mean a better rating always?



```
In [39]:
plt.figure(figsize=(10,5))

plt.scatter(dt3['Rating'], dt3['Reviews'], color='b', s=50 )

plt.xlabel ( " Rating " )
plt.ylabel ( " Reviews" )
plt.title ( " Rating vs. Reviews plot " )

# yes, reviews and rating have a strong positive correlation
```

4. Make boxplot for Rating vs. Content Rating
    0. Is there any difference in the ratings? Are some types liked better?

In [40]:

```
plt.figure(figsize=[15,7])
sns.boxplot("Rating","Content Rating",data=dt3)

#Mostly adults(18+) have always rated it high
```

5. Make boxplot for Ratings vs. Category
    0. Which genre has the best ratings? For each of the plots above, note down your observation.

In [41]:

```
plt.figure(figsize=[15,7])
sns.boxplot("Rating","Category",data=dt3)

# ART and Design genre has best rating
```

9. Data preprocessing

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

1. 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.

In [42]:

```
inp1=dt3.copy()
```

In [43]:

```
inp1['Reviews']=np.log1p(inp1['Reviews'])
```

In [44]:

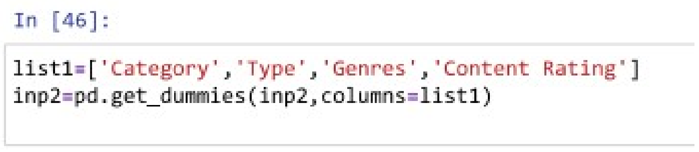
```
inp1['Installs']=np.log1p(inp1['Installs'])
```

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

In [45]:

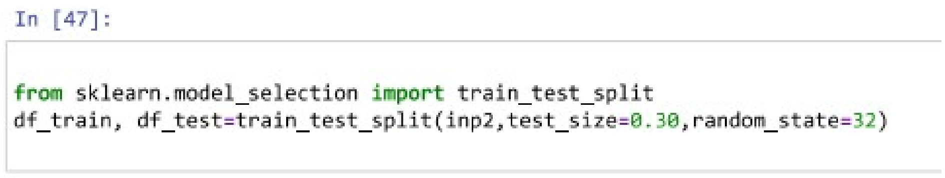
```
col_drop=['App','Last Updated','Current Ver', 'Android Ver']
inp2=inp1.drop(col_drop, axis=1).copy()
```

3. 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**.



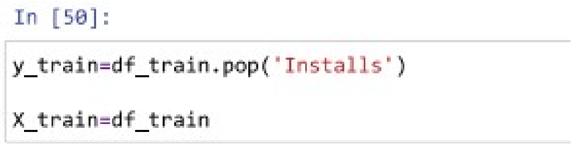
```
In [46]:
list1=['Category','Type','Genres','Content Rating']
inp2=pd.get_dummies(inp2,columns=list1)
```

10. Train test split and apply 70-30 split. Name the new dataframes df_train and df_test.

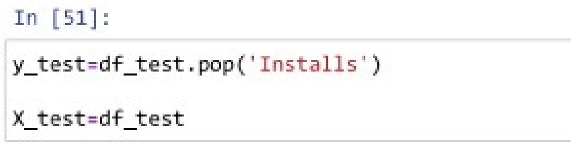


```
In [47]:

from sklearn.model_selection import train_test_split
df_train, df_test=train_test_split(inp2,test_size=0.30,random_state=32)
```

11. Separate the dataframes into X_train, y_train, X_test, and y_test.



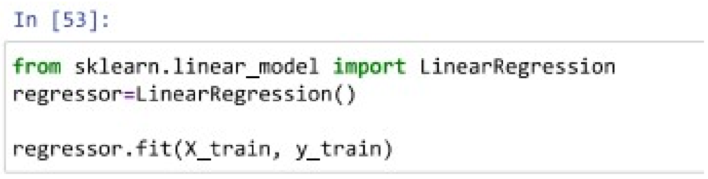
```
In [50]:
y_train=df_train.pop('Installs')

X_train=df_train
```



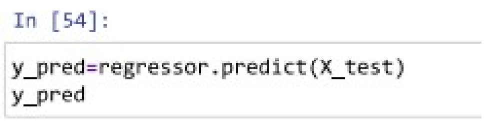
```
In [51]:
y_test=df_test.pop('Installs')

X_test=df_test
```

12 . Model building

- Use linear regression as the technique
- Report the R2 on the train set

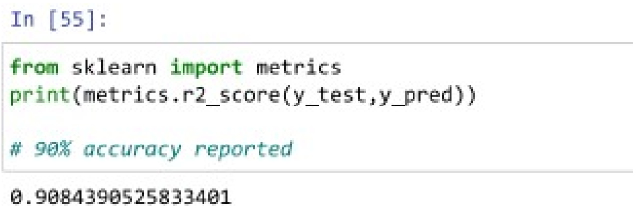


```
In [53]:
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()

regressor.fit(X_train, y_train)
```



```
In [54]:
y_pred=regressor.predict(X_test)
y_pred
```

13. Make predictions on test set and report R2.



```
In [55]:
from sklearn import metrics
print(metrics.r2_score(y_test,y_pred))

# 90% accuracy reported

0.9084390525833401
```