AppRatingProj

February 26, 2021

1 App Rating Prediction

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

1.0.2 >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"). boosting in search results visibility.

This feature will help bring more attention to newer apps that have the potential.

1.0.3 >Steps To Perform:

0. Load packages

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statistics as stc
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

1. Load the data file using pandas.

[3]:
```
df=pd.read_csv("googleplaystore.csv")
```

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

[4]:
```
print("Count of null values in data")
df.isnull().sum()
```

```
Count of null values in data
```

[4]:
```
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
```

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

[5]:
```
df.dropna(inplace=True)
print("Check for null values after removing nulls")
df.isnull().sum()
```

```
Check for null values after removing nulls
```

[5]:
```
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
```

4. Fixing Variables with inconsistent data 4.1 Size column has sizes in Kb as well as Mb. To analyze, you'll need to convert these to numeric

4.1.1 Extract the numeric value from the column

- Some values of size is not determined as it depends on device. To contiue such values will be droped

[6]:
```
df=df[-df['Size'].str.contains('Var')]
```

- There is a value with + sign should be handled

[7]:
```
df.loc[:,'SizeNum'] =df.Size.str.rstrip('Mk+')
df.SizeNum=pd.to_numeric(df['SizeNum'])
df.SizeNum.dtype
```

[7]:
```
dtype('float64')
```

4.1.2 Multiply the value by 1,000, if size is mentioned in Mb

[8]:
```
df['SizeNum']=np.where(df.Size.str.contains('M'),df.SizeNum*1000, df.SizeNum)
```

[9]:
```
# Size no more needed, replace it with SizeNum and drop SizeNum
df.Size=df.SizeNum
df.drop('SizeNum',axis=1,inplace=True)
#df
```

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

[10]:
```
df.Reviews = pd.to_numeric(df.Reviews)
```

[11]:
```
df.Reviews.dtype
```

[11]:
```
dtype('int64')
```

4.3 Installs field is currently stored as string and has values like 1,000,000+.

4.3.1 Treat 1,000,000+ as 1,000,000

[12]:
```
df['Installs']=df.Installs.str.replace("+","")
```

4.3.2 remove '+', ',' from the field, convert it to integer

[13]:
```
df.Installs=df.Installs.str.replace(",","")
df.Installs=pd.to_numeric(df.Installs)
df.Installs.dtype
```

[13]:
```
dtype('int64')
```

4.4 Price field is a string and has $ symbol. Remove '$' sign, and convert it to numeric.

[14]:
```
df.Price=df.Price.str.replace("$","")
df.Price=pd.to_numeric(df.Price)
df.Price.dtype
```

[14]:
```
dtype('float64')
```

4.5.1 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.

[15]:
```
df=df[(df.Rating>=1) & (df.Rating<=5) ]
```

4.5.2 Reviews should not be more than installs as only those who installed can review the app. If there are any such records, drop them.

[16]:
```
len(df.index)
```

[16]:
```
7723
```

[17]:
```
df.drop(df.index[df.Reviews>df.Installs],axis=0,inplace=True)
len(df.index)
```

[17]: 7717

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

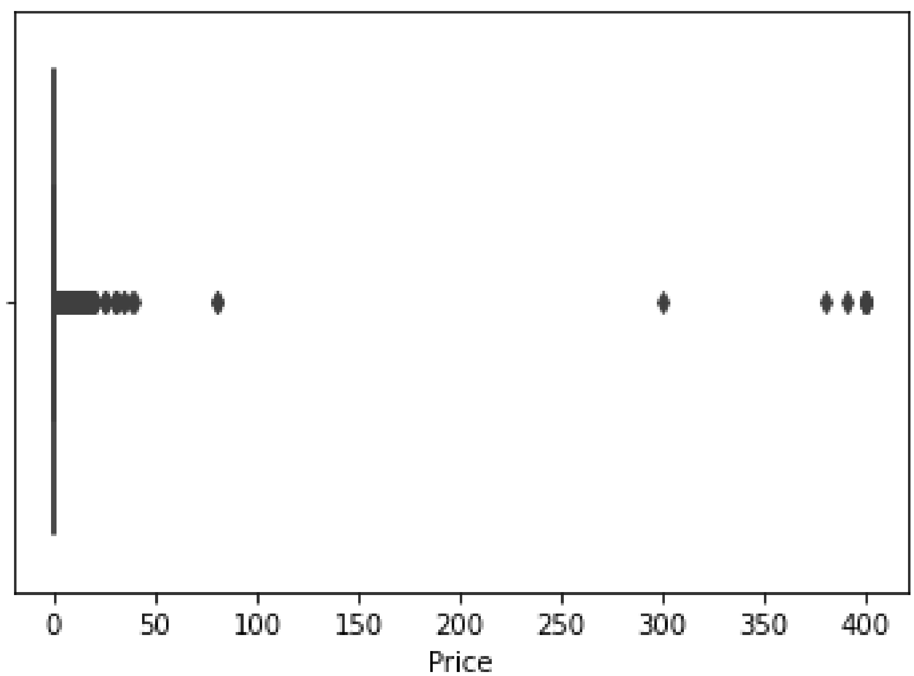
[18]:
```
index_free_and_price_gt_0=df.index[((df.Type=='Free')&(df.Price>0))]
if len(index_free_and_price_gt_0)>0:
    print("Dropping following indices:",index_free_and_price_gt_0)
    df.drop(index_free_and_price_gt_0,axis=0,inplace=True)
else:
    print("There is no Free Apps with price >0")
```

```
There is no Free Apps with price >0
```

5.Performing univariate analysis: 5.1 Boxplot for Price

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

[19]:
```
ax = sns.boxplot(x='Price', data=df)
```



- Insights: Most of Price values are less than 50 while there is some near concentration around 80. greater than 100 may be considered outliers
- Consider 3 STD as range of outliers

[20]:
```
price_std=stc.stdev(df.Price)
price_std
```

[20]: 17.414783874309933

[21]:
```
price_mean=stc.mean(df.Price)
price_mean
```

[21]: 1.128724893093171

[22]:
```
price_outlier_uplimit=price_mean+3*price_std
price_outlier_uplimit
```

[22]: 53.37307651602297

[23]:
```
#price_outlier_downlimit=price_mean-3*price_std
#price_outlier_downlimit
```

[24]:
```
#df[df.Price>price_outlier_uplimit]
print("# of upper outliers is ",len(df[(df.Price>price_outlier_uplimit) ]))
```

```
# of upper outliers is  17
```

[25]:
```
#df[df.Price<price_outlier_downlimit]
#print("# of lower outliers is ",len(df[df.Price<price_outlier_downlimit]))
```

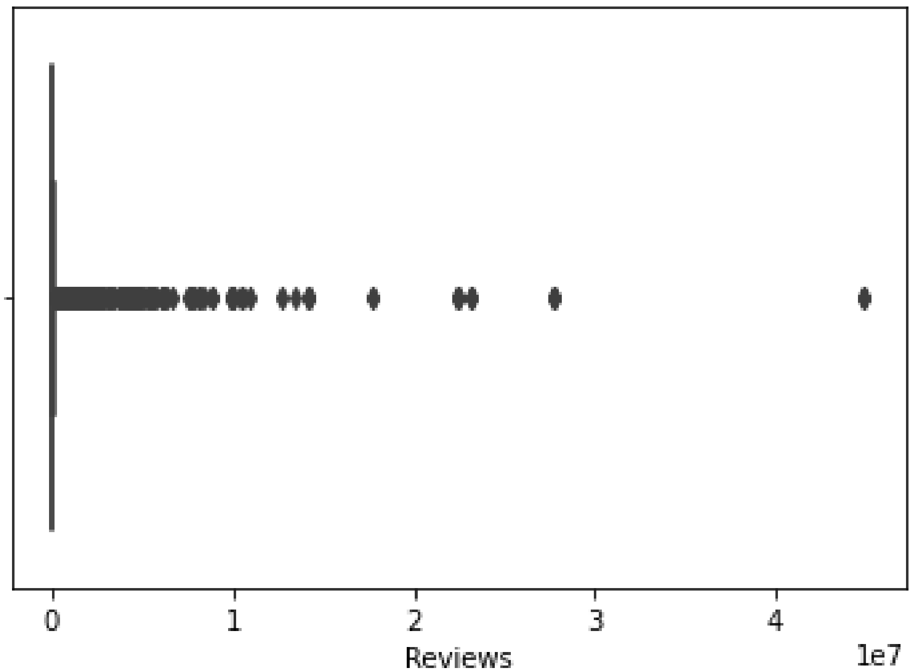
- It seems there are about 17 outliers

5.2 Boxplot for Reviews

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

[26]:
```
sns.boxplot(x='Reviews',data=df)
```

[26]: <AxesSubplot:xlabel='Reviews'>



- Insights: Most Apps get about more than 2M review. Roughly, greater than 2M can be considered outliers
- Consider 3 STD as range of outliers

[27]:
```
rev_std=stc.stdev(df.Reviews)
rev_std
```

[27]: 1864639.6094670836

[28]:
```
rev_mean=stc.mean(df.Reviews)
rev_mean
```

[28]: 295127.5482700531

[29]:
```
rev_outlier_uplimit=rev_mean+3*rev_std
rev_outlier_uplimit
```

[29]: 5889046.376671304

[30]:
```
rev_outlier_downlimit=rev_mean-3*rev_std
rev_outlier_downlimit
```

[30]: -5298791.280131198

[31]:
```
#df[df.Reviews>rev_outlier_uplimit]
print("# of upper outliers is ",len(df[(df.Reviews>rev_outlier_uplimit) ]))
```

```
# of upper outliers is  89
```

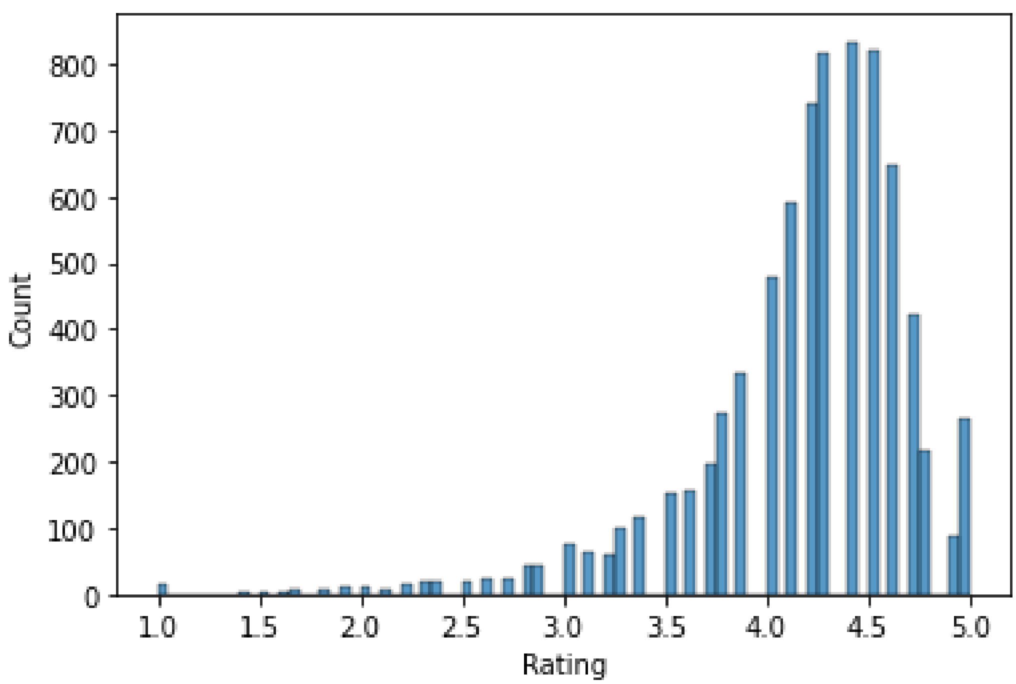
[32]:
```
# Since reviews cannot be less than 1, no need to check lower outliers
# remove outliers
#df.drop(df.index[(df.Reviews>rev_outlier_uplimit) ],inplace=True)
#len(df.index)
```

5.3 Histogram for Rating

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

[33]:
```
#sns.boxplot(x='Rating',data=df)
sns.histplot(x='Rating',data=df)
```

[33]:
```
<AxesSubplot:xlabel='Rating', ylabel='Count'>
```



- Rating tends to higher values

[34]:
```
#rating_std=stc.stdev(df.Rating)
#rating_std
```

[35]:
```
#rating_mean=stc.mean(df.Rating)
#rating_mean
```

[36]:
```
#rating_outlier_uplimit=rating_mean+3*rating_std
#rating_outlier_uplimit
```

[37]:
```
#rating_outlier_downlimit=rating_mean-3*rating_std
#rating_outlier_downlimit
```

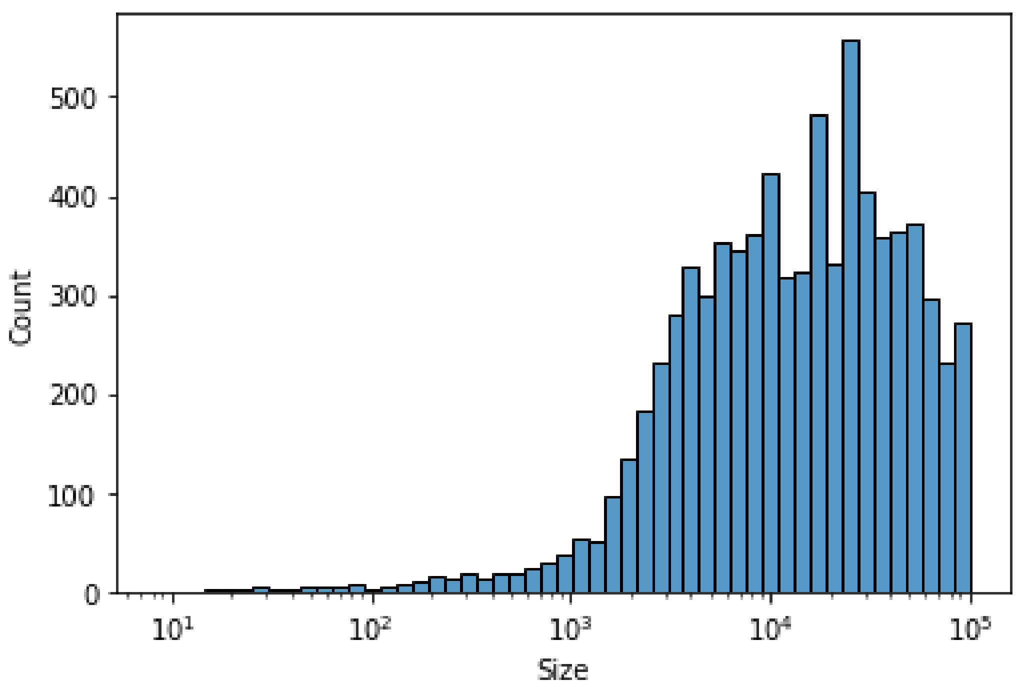
[38]:
```
# Since max possible value of rating (5) is less than upper limit, no need to␣
↪manage upper outliers
#df[df.Rating<rating_outlier_downlimit]
#print("# of lower outliers is ",len(df[(df.Rating<rating_outlier_downlimit) ]))
```

[39]:
```
#df.drop(df.index[(df.Rating<rating_outlier_downlimit)],inplace=True)
#len(df.index)
```

5.4 Histogram for Size

[40]:
```
# use log scale to make histogram more representable
sns.histplot(x='Size',data=df,log_scale=True)
```

[40]: <AxesSubplot:xlabel='Size', ylabel='Count'>



6. Outlier treatment: 6.1 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!

6.1.1 Check out the records with very high price

[41]:

```python
df[df.Price>=200]
```

[41]:

```
                              App   Category  Rating  Reviews     Size  \
4197          most expensive app (H)     FAMILY     4.3        6   1500.0
4362                        I'm rich  LIFESTYLE     3.8      718  26000.0
4367        I'm Rich - Trump Edition  LIFESTYLE     3.6      275   7300.0
5351                        I am rich  LIFESTYLE     3.8     3547   1800.0
5354                   I am Rich Plus     FAMILY     4.0      856   8700.0
5355                    I am rich VIP  LIFESTYLE     3.8      411   2600.0
5356                I Am Rich Premium    FINANCE     4.1     1867   4700.0
5357              I am extremely Rich  LIFESTYLE     2.9       41   2900.0
5358                       I am Rich!    FINANCE     3.8       93  22000.0
5359                 I am rich(premium)    FINANCE     3.5      472    965.0
5362                    I Am Rich Pro     FAMILY     4.4      201   2700.0
5364  I am rich (Most expensive app)    FINANCE     4.1      129   2700.0
5366                        I Am Rich     FAMILY     3.6      217   4900.0
5369                        I am Rich    FINANCE     4.3      180   3800.0
5373                I AM RICH PRO PLUS    FINANCE     4.0       36  41000.0

      Installs  Type   Price Content Rating         Genres      Last Updated  \
4197       100  Paid  399.99       Everyone  Entertainment     July 16, 2018
4362     10000  Paid  399.99       Everyone      Lifestyle    March 11, 2018
4367     10000  Paid  400.00       Everyone      Lifestyle       May 3, 2018
5351    100000  Paid  399.99       Everyone      Lifestyle  January 12, 2018
5354     10000  Paid  399.99       Everyone  Entertainment      May 19, 2018
5355     10000  Paid  299.99       Everyone      Lifestyle     July 21, 2018
5356     50000  Paid  399.99       Everyone        Finance November 12, 2017
5357      1000  Paid  379.99       Everyone      Lifestyle      July 1, 2018
5358      1000  Paid  399.99       Everyone        Finance December 11, 2017
5359      5000  Paid  399.99       Everyone        Finance       May 1, 2017
5362      5000  Paid  399.99       Everyone  Entertainment      May 30, 2017
5364      1000  Paid  399.99           Teen        Finance  December 6, 2017
5366     10000  Paid  389.99       Everyone  Entertainment     June 22, 2018
5369      5000  Paid  399.99       Everyone        Finance    March 22, 2018
5373      1000  Paid  399.99       Everyone        Finance     June 25, 2018

     Current Ver   Android Ver
4197         1.0    7.0 and up
4362       1.0.0    4.4 and up
4367       1.0.1    4.1 and up
5351         2.0  4.0.3 and up
5354         3.0    4.4 and up
5355       1.1.1    4.3 and up
5356         1.6    4.0 and up
5357         1.0    4.0 and up
5358         1.0    4.1 and up
5359         3.4    4.4 and up
```

```
5362        1.54    1.6 and up
5364           2  4.0.3 and up
5366         1.5    4.2 and up
5369         1.0    4.2 and up
5373       1.0.2    4.1 and up
```

[42]:
```
print("# of Apps with price >= 200 = ",len(df[(df.Price>=200) ]))
```

```
# of Apps with price >= 200 =  15
```

6.1.1.1 Is 200 indeed a high price? It is very high and very far than the mean

6.1.2 Drop these as most seem to be junk apps

[43]:
```
df.drop(df.index[(df.Price>=200)], inplace=True)
len(df.index)
```

[43]: 7702

6.2 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.

[44]:
```
df.drop(df.index[(df.Reviews>=2000000)], inplace=True)
len(df.index)
```

[44]: 7483

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

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

[45]:
```
install_10_perc=np.percentile(df.Installs, 10)
install_10_perc
```

[45]: 1000.0

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

[46]:
```
install_25_perc=np.percentile(df.Installs, 25)
install_25_perc
```

[46]: 10000.0

[47]:
```
install_50_perc=np.percentile(df.Installs, 50)
install_50_perc
```

[47]: 100000.0

[48]:
```
install_70_perc=np.percentile(df.Installs, 70)
install_70_perc
```

[48]: 1000000.0

[49]:
```
install_90_perc=np.percentile(df.Installs,90)
install_90_perc
```

[49]: 10000000.0

[50]:
```
install_95_perc=np.percentile(df.Installs,95)
install_95_perc
```

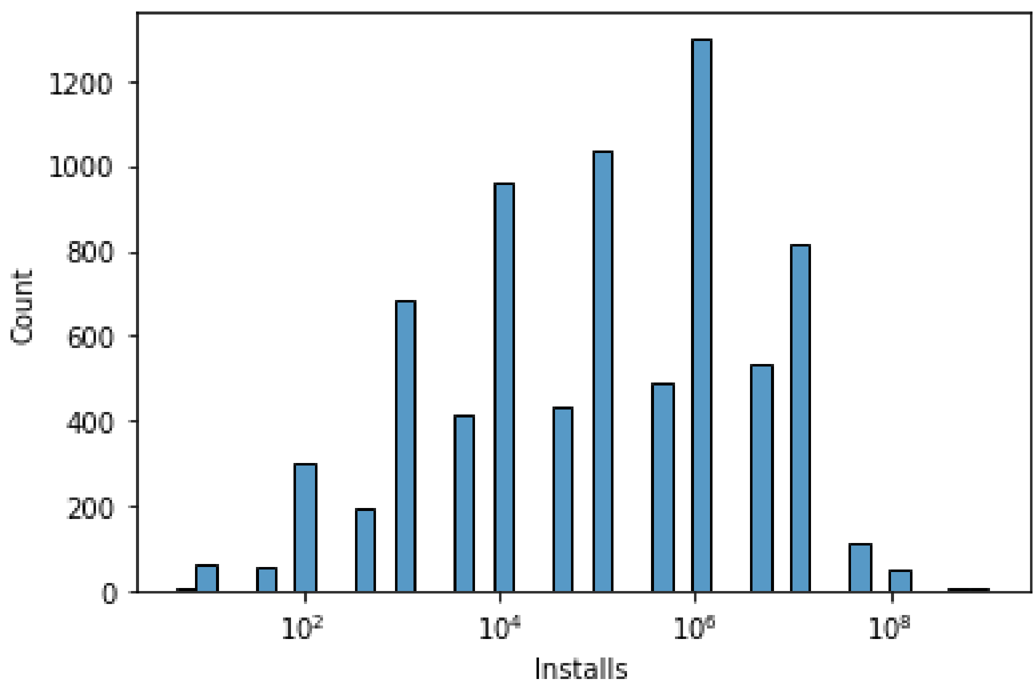
[50]: 10000000.0

[51]:
```
install_99_perc=np.percentile(df.Installs,99)
install_99_perc
```

[51]: 50000000.0

[52]:
```
sns.histplot(data=df,x='Installs',log_scale=True)
```

[52]: <AxesSubplot:xlabel='Installs', ylabel='Count'>



- My decision is to drop values > percentile of 99(Almost 3 STD)

[53]:
```
print("As result, ",len(df[df.Installs >= install_99_perc])," will be dropped")
```

```
As result,  176  will be dropped
```

[54]:
```
df.drop(df.index[df.Installs >= install_99_perc],inplace=True)
len(df.index)
```

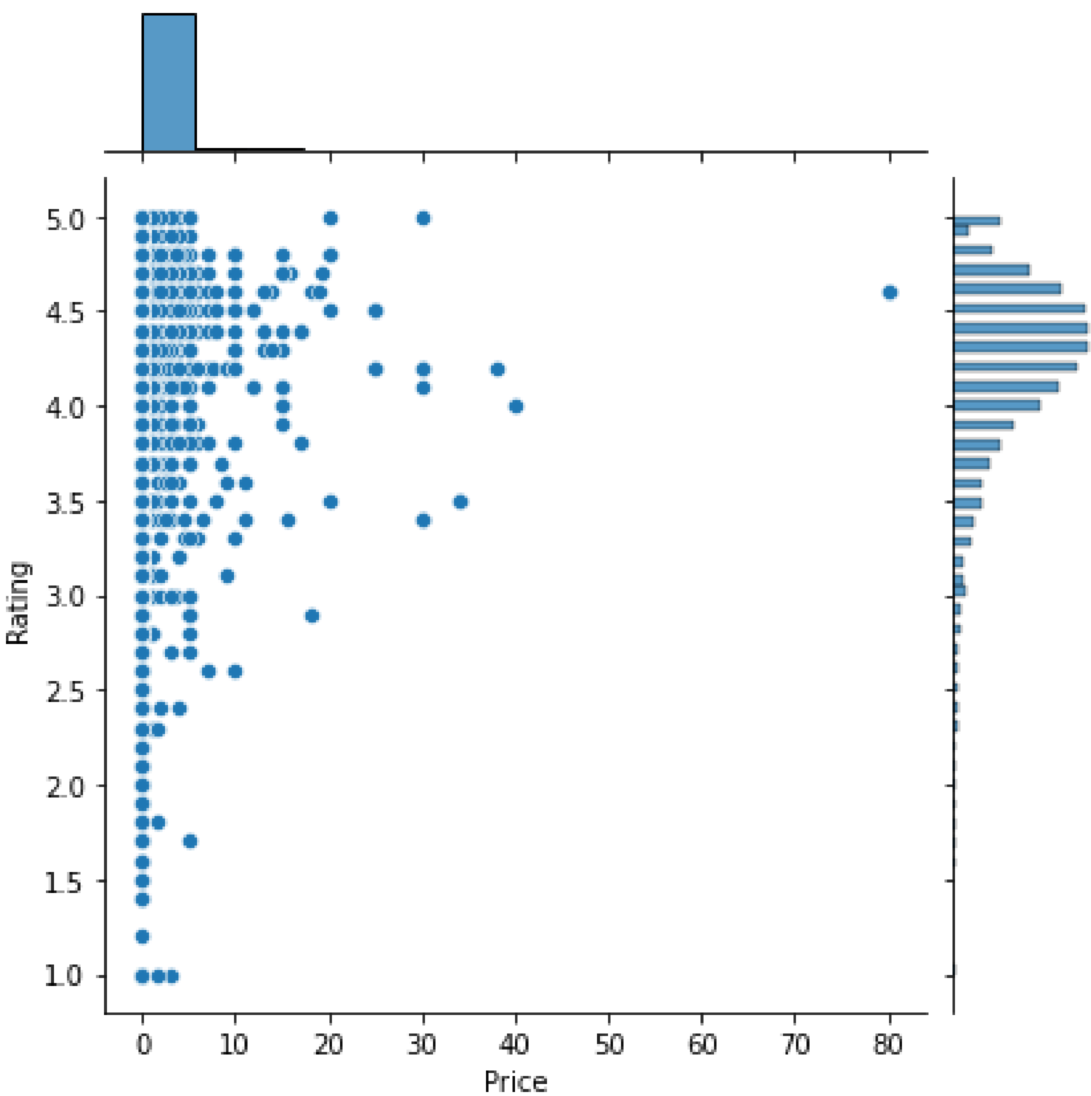
[54]: 7307

7. 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 For each of the plots , note down your observation.

7.1. Make scatter plot/joinplot for Rating vs. Price

[55]:
```
sns.jointplot(data=df,y='Rating',x='Price')
```

[55]: <seaborn.axisgrid.JointGrid at 0x12caa27f490>



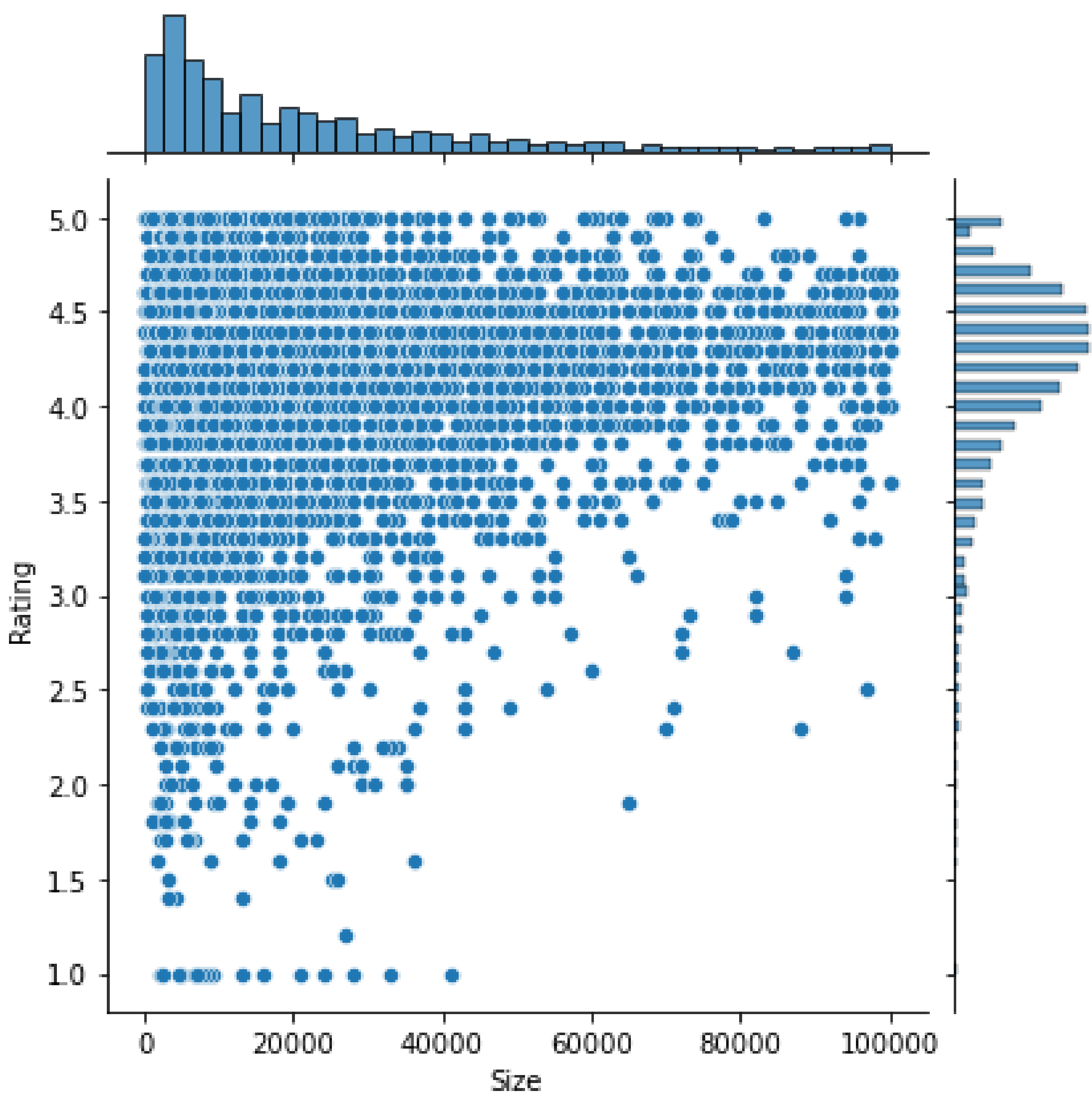
7.2. What pattern do you observe? Does rating increase with price?

Most of Apps with high price get > 3 Rating but this is because majority of apps are with low price. In addition most apps get rating > 3. Concusion: We cannot consider there is a good relationship between Rating and Price. It seems Price has limited impact on Rating.

7.3. Make scatter plot/joinplot for Rating vs. Size

[56]:
```
sns.jointplot(data=df,y='Rating',x='Size')
```

[56]:
```
<seaborn.axisgrid.JointGrid at 0x12cab43d250>
```



7.4. Are heavier apps rated better?

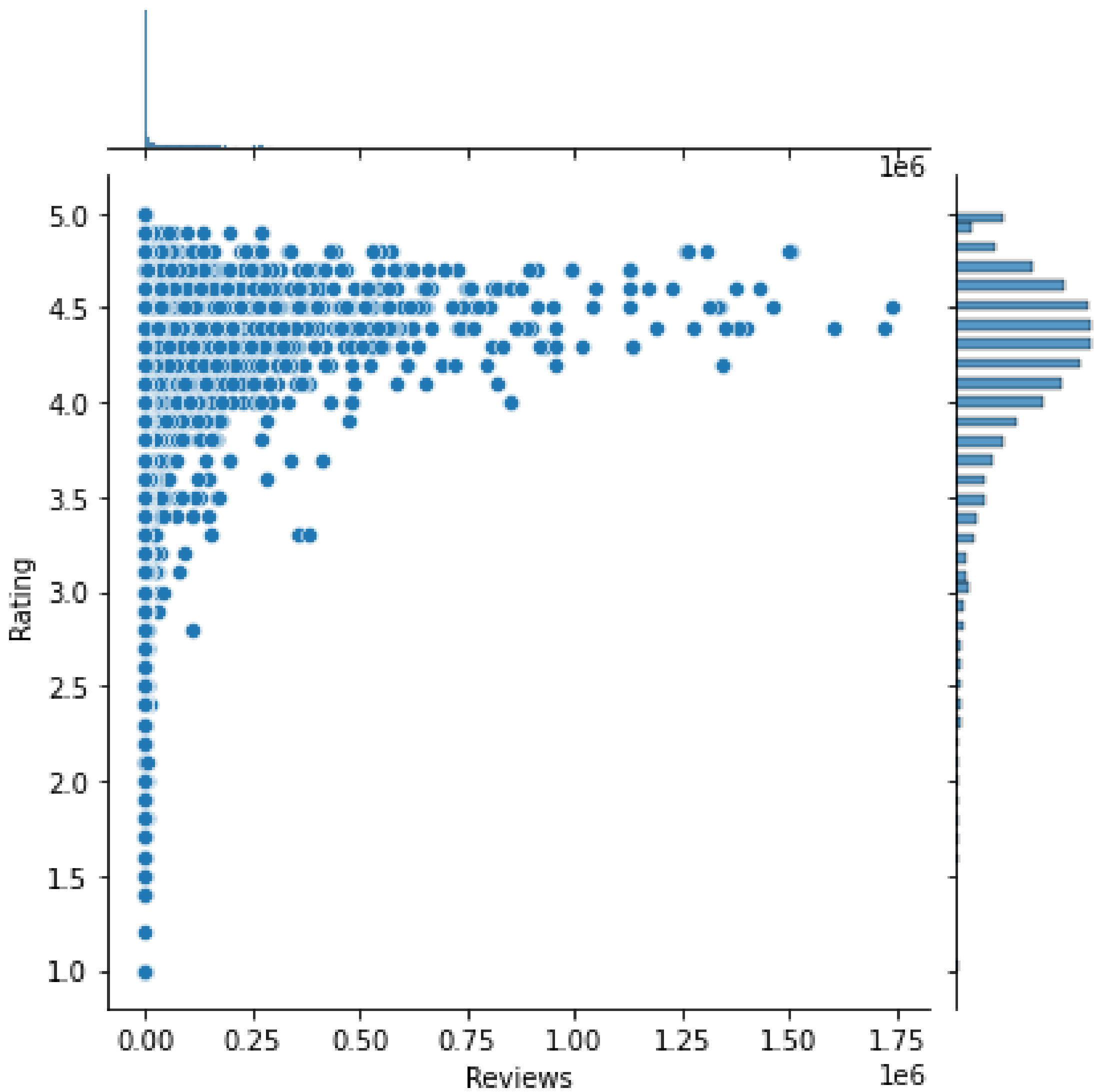
Again if we look to the area where most apps rated (greater than 3) almost the points are evenly distributed

The relationship between Size and rating is very weak

7.5. Make scatter plot/joinplot for Rating vs. Reviews

[57]:
```
sns.jointplot(data=df,y='Rating',x='Reviews')
```

[57]: <seaborn.axisgrid.JointGrid at 0x12caa2b84f0>



7.6. Does more review mean a better rating always?

Although the relationship seems also not so strong, but we can notice that there is some concentration of apps with higher reviews in high rating area. It seems good apps get more reviews than others

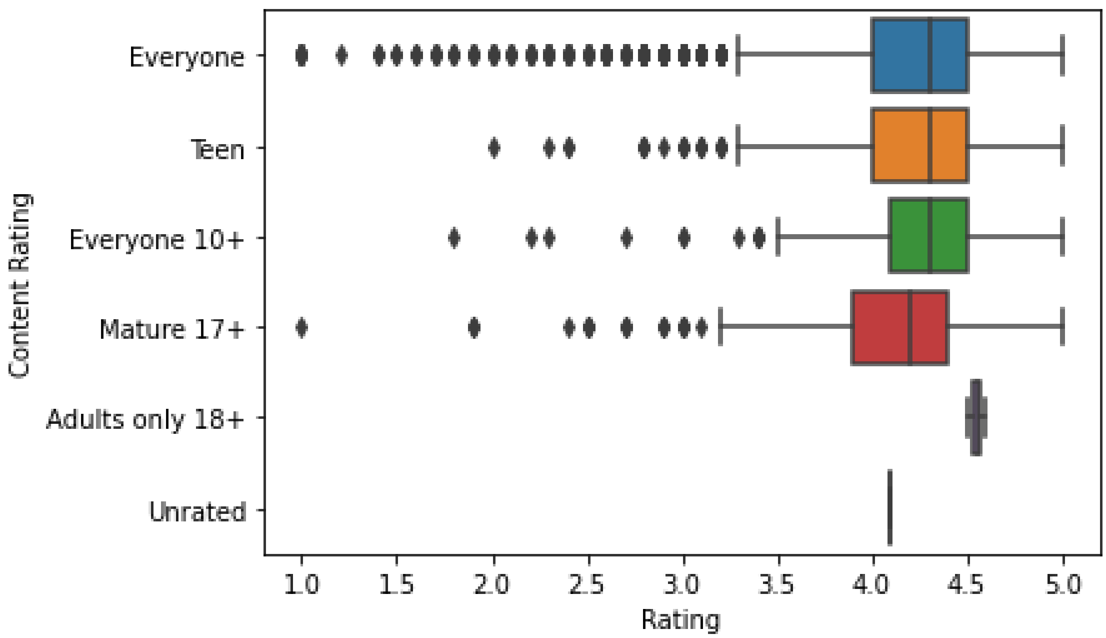
7.7 Make boxplot for Rating vs. Content Rating

[58]:
```
df['Content Rating'].unique()
```

[58]:
```
array(['Everyone', 'Teen', 'Everyone 10+', 'Mature 17+',
       'Adults only 18+', 'Unrated'], dtype=object)
```

[59]:
```
sns.boxplot(data=df,x='Rating',y='Content Rating')
```

[59]:
```
<AxesSubplot:xlabel='Rating', ylabel='Content Rating'>
```



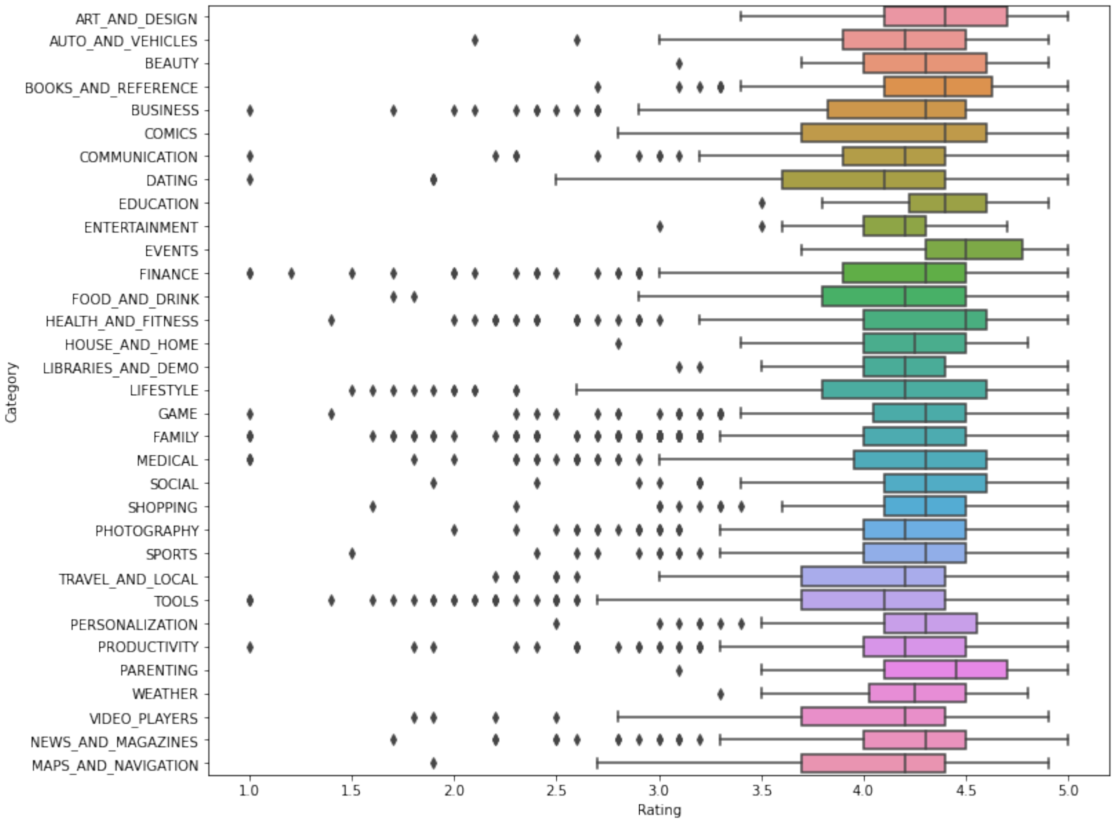
7.8. Is there any difference in the ratings? Are some types liked better?

Apps of Adults only 18+ has higher rating than others while Mature 17+ gets less liks. Others seem to be closed. Content has good impact on Rating

7.9. Make boxplot for Ratings vs. Category

[60]:
```
a4_dims = (11.7, 10.27)
fig, ax = plt.subplots(figsize=a4_dims)
sns.boxplot(data=df,x='Rating',y='Category',ax=ax)
```

[60]:
```
<AxesSubplot:xlabel='Rating', ylabel='Category'>
```



7.10. Which genre has the best ratings?

The best genre is Events

8. 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.
2. Drop columns App, Last Updated, Current Ver, and Android Ver. These variables are not useful for our task.
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.

[61]:
```
#8.1
inp1=df.copy()
inp1.Reviews=inp1.Reviews.apply(np.log1p)
```

[62]:
```
inp1.Installs=inp1.Installs.apply(np.log1p)
```

[63]:
```
#8.2
inp1.drop(columns=['App','Last Updated','Current Ver','Android␣
↪Ver'],inplace=True)
```

[64]:
```
inp1.shape
```

[64]: (7307, 9)

[65]:
```
#8.3
inp2= pd.get_dummies(inp1)
```

[66]:
```
inp2.shape
```

[66]: (7307, 158)

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

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

[67]:
```
data = inp2.drop(columns='Rating')
data.shape
```

[67]: (7307, 157)

[68]:
```
target = pd.DataFrame(inp2.Rating)
target.shape
```

[68]: (7307, 1)

[69]:
```
x_train, x_test, y_train, y_test = train_test_split(data, target, test_size=0.
↪3, random_state=3)
print("x_train shape is ", x_train.shape)
print("y_train shape is ", y_train.shape)
print("x_test shape is ", x_test.shape)
print("y_test shape is ", y_test.shape)
```

```
x_train shape is  (5114, 157)
y_train shape is  (5114, 1)
x_test shape is  (2193, 157)
y_test shape is  (2193, 1)
```

11. Model building Use linear regression as the technique

Report the R2 on the train set

[70]:
```
model=LinearRegression()
model.fit(x_train, y_train)
```

[70]:
```
LinearRegression()
```

[71]:
```
train_pred=model.predict(x_train)
```

[72]:
```
print("R2 value of the model(by train) is ", r2_score(y_train, train_pred))
```

```
R2 value of the model(by train) is  0.15264772134593896
```

12. Make predictions on test set and report R2.

[73]:
```
test_pred=model.predict(x_test)
```

[74]:
```
print("R2 value of the model(by test) is ", r2_score(y_test, test_pred))
```

```
R2 value of the model(by test) is  0.14262263030964129
```