App Rating Prediction

October 22, 2021

[1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Load the data file using pandas.

[2]:
```python
df = pd.read_csv(r'googleplaystore.csv')
```

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

[3]:
```python
df.isna().sum()
```

[3]:
```
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
```

Drop records with nulls in any of the columns.

[4]:
```python
df.dropna(axis=0, inplace=True)
```

Variables seem to have incorrect type and inconsistent formatting. You need to fix them:

Size column has sizes in Kb as well as Mb. To analyze, you'll need to convert these to numeric.

Extract the numeric value from the column

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

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

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

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

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

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

[5]:
```
# 'Varies with device' value is inappropriate for Size so drop it
df = df[df.Size != 'Varies with device']
```

[6]:
```
# Extract the numeric value from the column, Multiply the value by 1,000, if␣
↪size is mentioned in Mb
df.Size = df.Size.apply(lambda x: float(x[:-1])*1000 if x[-1] == 'M' else␣
↪float(x[:-1]))
```

[7]:
```
# Reviews is a numeric field that is loaded as a string field. Convert it to␣
↪numeric (int/float)
df.Reviews = df.Reviews.astype('int')
```

[8]:
```
# Installs field is currently stored as string and has values like 1,000,000+.
# Treat 1,000,000+ as 1,000,000
# remove ‘+’, ‘,’ from the field, convert it to integer

df.Installs = df.Installs.apply(lambda x: x[:-1] if x[-1] == '+' else x).apply(
                                        lambda x: ''.join(x.split(','))).astype('int')
```

[9]:
```
# Price field is a string and has     .    ‘’ sign, and convert it to numeric
df.Price = df.Price.apply(lambda x: x[1:] if x.startswith('$') else x).
↪astype('float')
```

Sanity checks:

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.

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

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

[10]:
```
df = df[(df.Rating >=1)&(df.Rating <=5) & (df.Reviews <=df.Installs) & ~((df.
↪Type == 'Free') & (df.Price >0))]
```

Performing univariate analysis:

Boxplot for Price

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

Boxplot for Reviews

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

Histogram for Rating

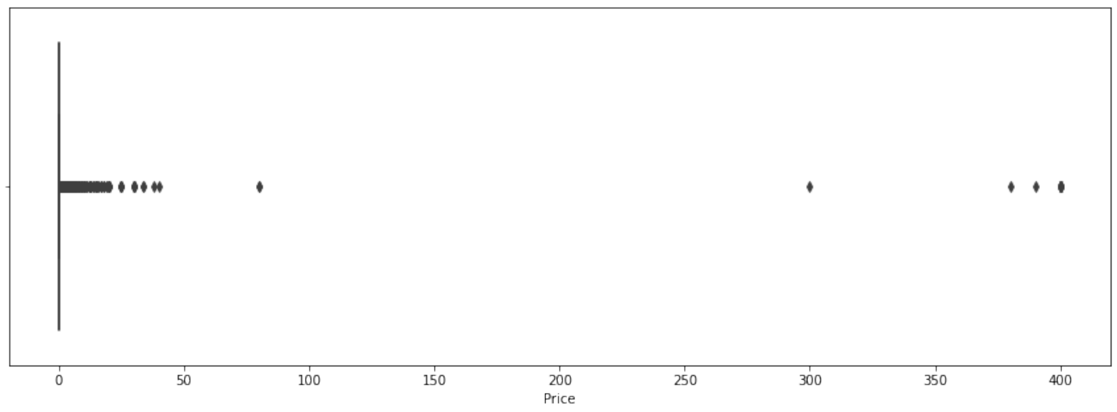
How are the ratings distributed? Is it more toward higher ratings?

Histogram for Size

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

```
[11]: # Boxplot for Price
      # Are there any outliers? Think about the price of usual apps on Play Store.
      plt.figure(figsize=(15,5))
      sns.boxplot(x= 'Price' , data=df)
```

```
[11]: <AxesSubplot:xlabel='Price'>
```

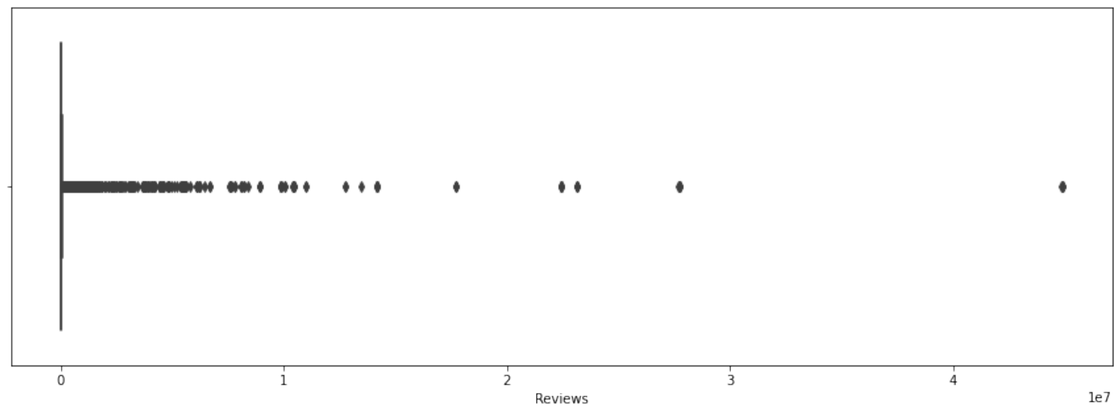


Are there any outliers? Think about the price of usual apps on Play Store.

Since the majority of Apps on playstore are Free, even the apps having very low price is treated as outlier.

```
[12]: # Boxplot for Reviews
      # Are there any apps with very high number of reviews? Do the values seem right?
      plt.figure(figsize=(15,5))
      sns.boxplot(x='Reviews',data=df)
```

```
[12]: <AxesSubplot:xlabel='Reviews'>
```

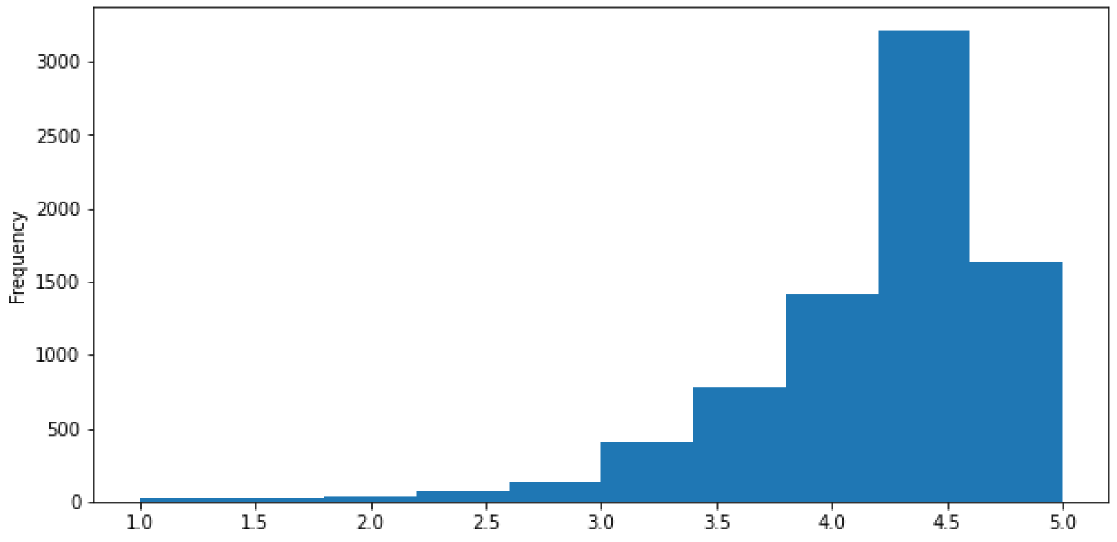


Are there any apps with very high number of reviews? Do the values seem right?

Yes, very few apps have very high number of reviews. Since these apps are very few and have very high number of reviews they can skew the result. Hence we can treat these reviews as outlier.

[13]:
```
# Histogram for Rating
# How are the ratings distributed? Is it more toward higher ratings?
plt.figure(figsize=(10,5))
df.Rating.plot.hist()
```

[13]:
```
<AxesSubplot:ylabel='Frequency'>
```

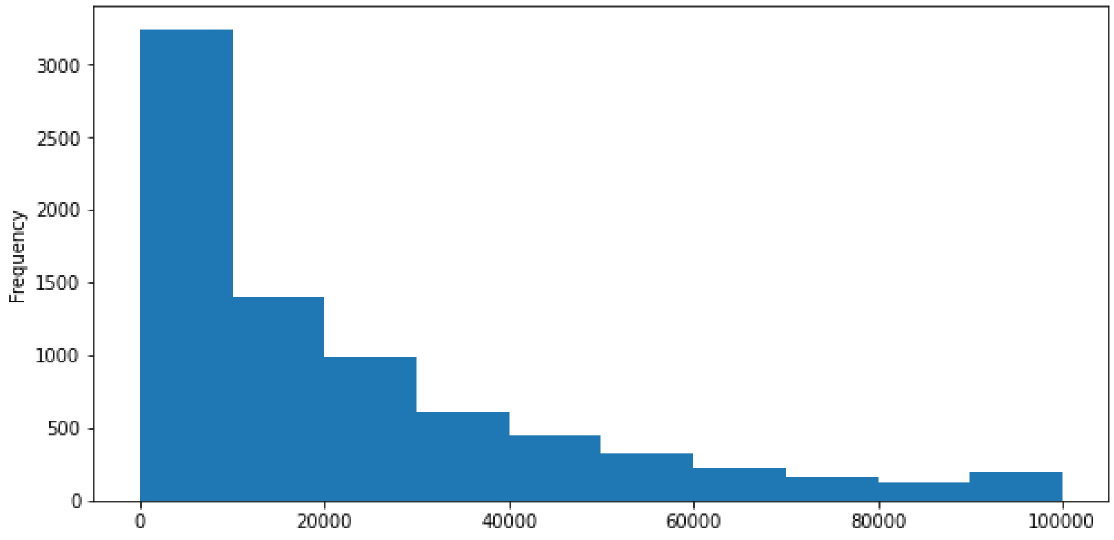


How are the ratings distributed? Is it more toward higher ratings?

The highest frequency of rating on playstore is 4.5. So we can say on playstore most of the app has 4.5 rating. Another conclusion we can draw here that on play store most of the app has rating between 3.8 to 5.0.

[14]:
```
# Histogram for Size
plt.figure(figsize=(10,5))
df.Size.plot.hist()
```

[14]:
```
<AxesSubplot:ylabel='Frequency'>
```



Most of the app on playstore has size lest than 10000kb.

Outlier treatment:

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!

Check out the records with very high price

Is 200 indeed a high price?

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.

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

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

[15]:
```
df.Price.quantile([.1, .25, .5, .7, .9, .95, .99])
```

[15]:
```
0.10    0.00
0.25    0.00
0.50    0.00
```

```
0.70    0.00
0.90    0.00
0.95    1.99
0.99    9.99
Name: Price, dtype: float64
```

If we notice the Price percentile. 99 percentile has value 9.99. It means 99 percent of app price is less than 10. So we can conclude that the price 200 is still very high.

[16]:
```
df = df[(df.Price < 10) & (df.Reviews < 2000000)]
```

[17]:
```
df.Installs.quantile([.1, .25, .5, .7, .8, .9, .95, .99])
```

[17]:
```
0.10        1000.0
0.25       10000.0
0.50      100000.0
0.70     1000000.0
0.80     5000000.0
0.90    10000000.0
0.95    10000000.0
0.99    50000000.0
Name: Installs, dtype: float64
```

[18]:
```
df = df[df.Installs <= 1000000]
```

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.

Make scatter plot/joinplot for Rating vs. Price

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

Make scatter plot/joinplot for Rating vs. Size

Are heavier apps rated better?

Make scatter plot/joinplot for Rating vs. Reviews

Does more review mean a better rating always?

Make boxplot for Rating vs. Content Rating

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

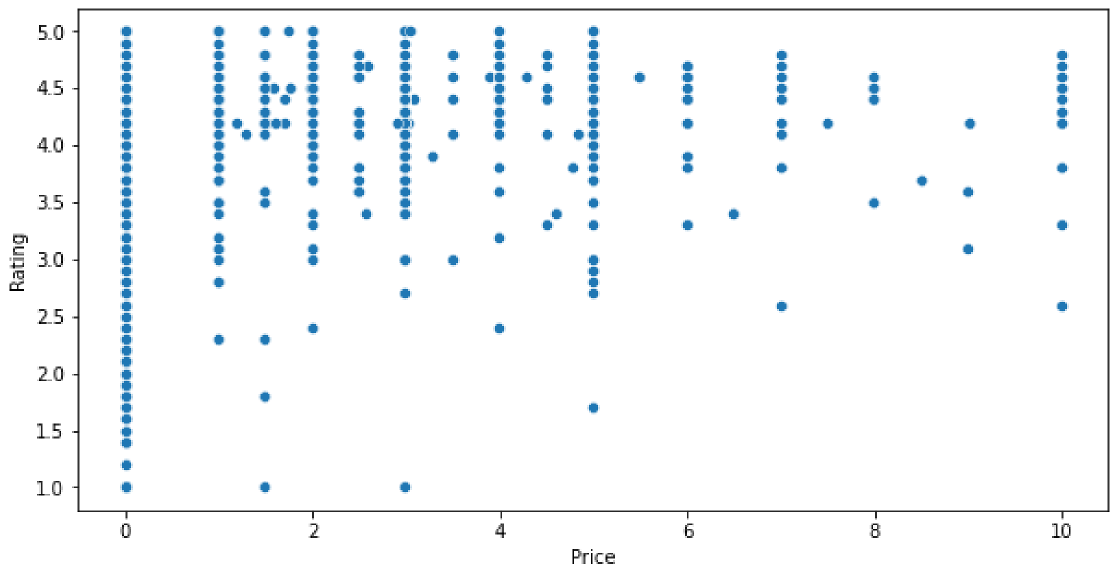
Make boxplot for Ratings vs. Category

Which genre has the best ratings?

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

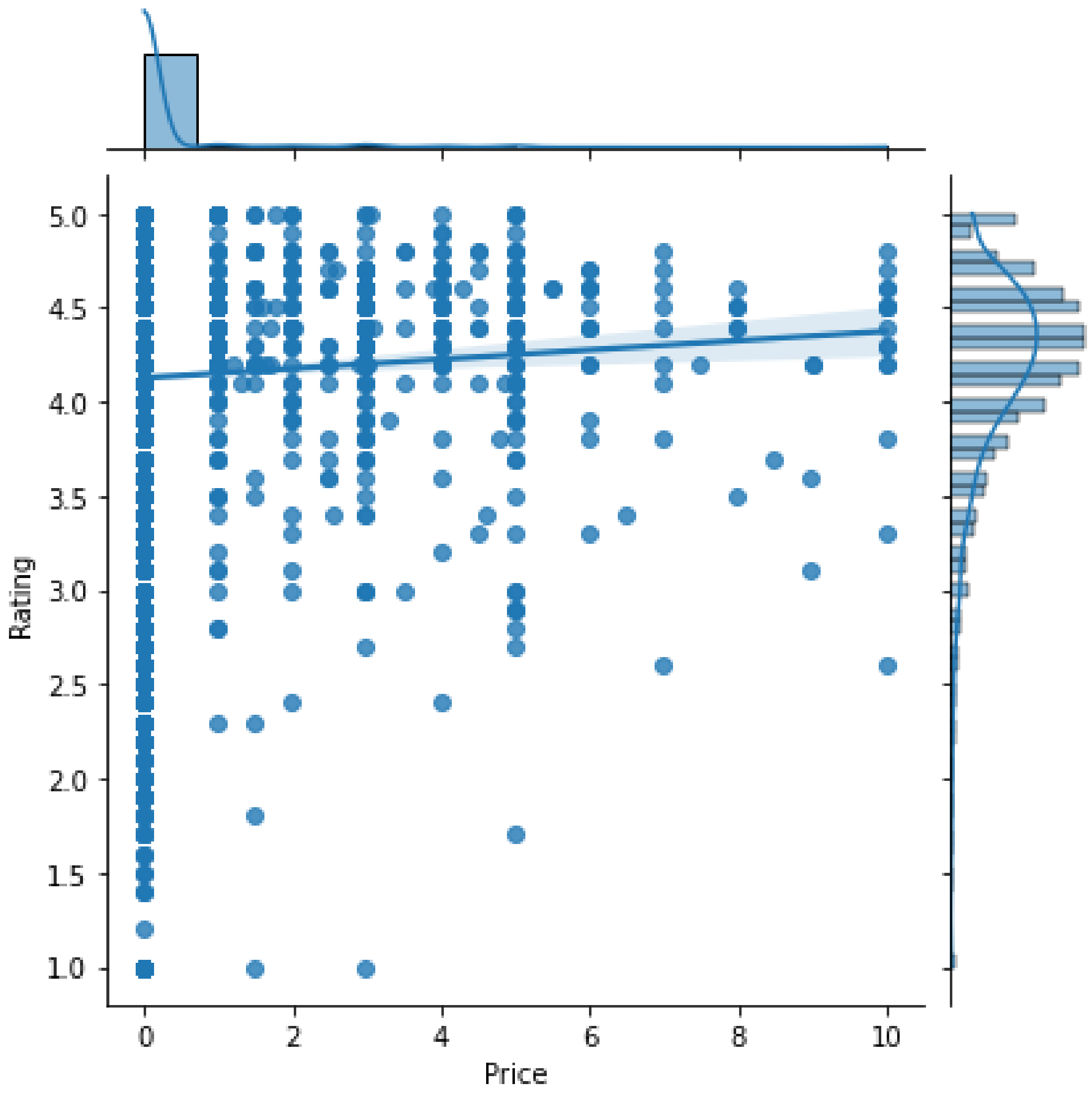
[19]:
```
# Make scatter plot/joinplot for Rating vs. Price
# What pattern do you observe? Does rating increase with price?
plt.figure(figsize=(10,5))
sns.scatterplot(x='Price', y='Rating', data=df)
```

[19]:
```
<AxesSubplot:xlabel='Price', ylabel='Rating'>
```



[20]:
```
# plt.figure(figsize=(20,5))
sns.jointplot(x='Price', y='Rating', data=df, kind='reg')
```

[20]:
```
<seaborn.axisgrid.JointGrid at 0x21fe9c65dc0>
```

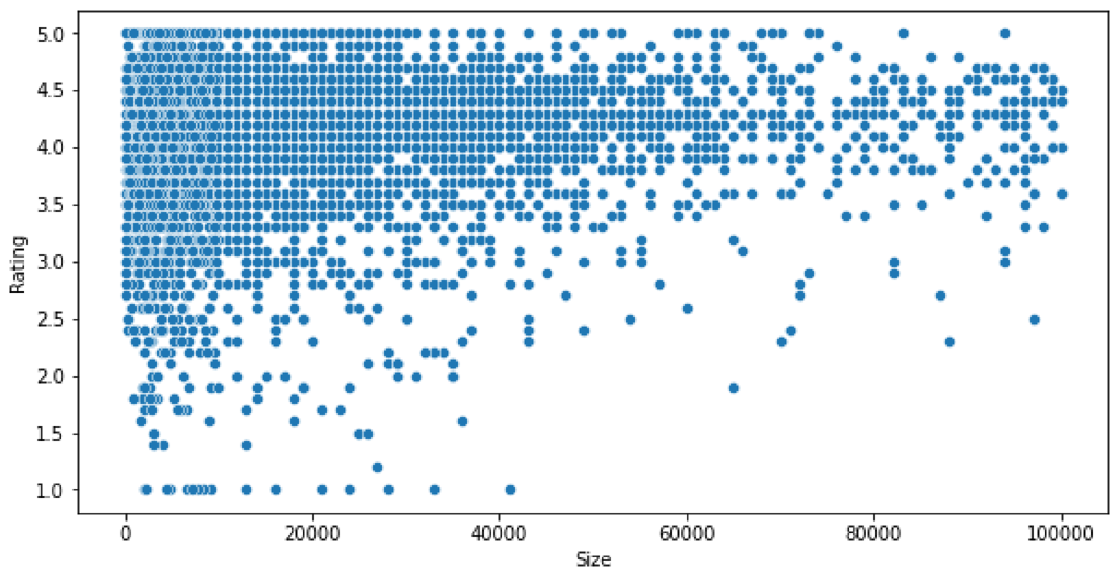


What pattern do you observe? Does rating increase with price?

There is slight increase in rating with increase in Price

[21]:
```
# Make scatter plot/joinplot for Rating vs. Size
# Are heavier apps rated better?
plt.figure(figsize=(10,5))
sns.scatterplot(x='Size', y='Rating', data=df)
```

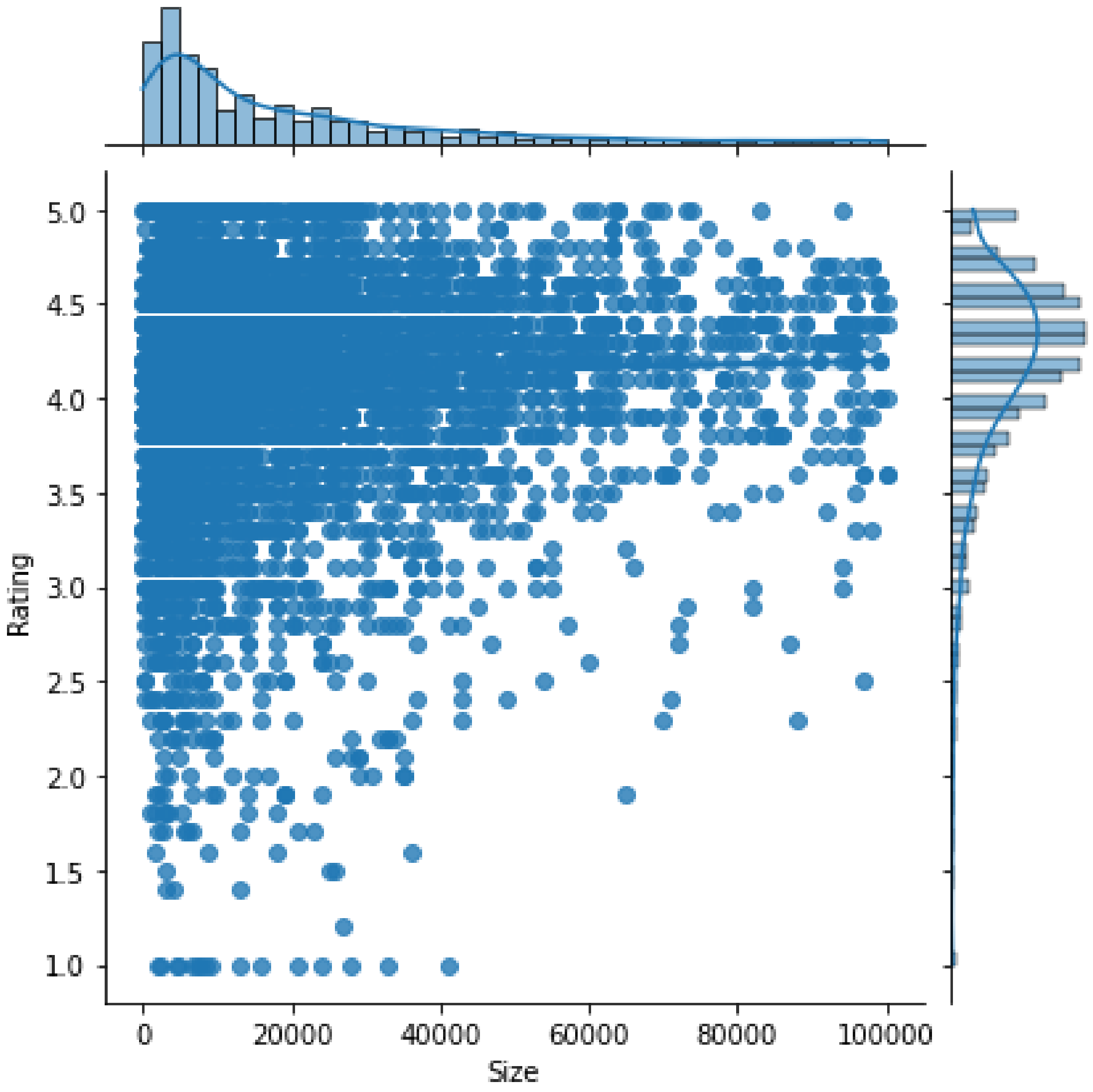
[21]:
```
<AxesSubplot:xlabel='Size', ylabel='Rating'>
```



[22]:
```
plt.figure(figsize=(10,5))
sns.jointplot(x='Size', y='Rating', data=df, kind='reg')
```

[22]:
```
<seaborn.axisgrid.JointGrid at 0x21febea7d30>

<Figure size 720x360 with 0 Axes>
```

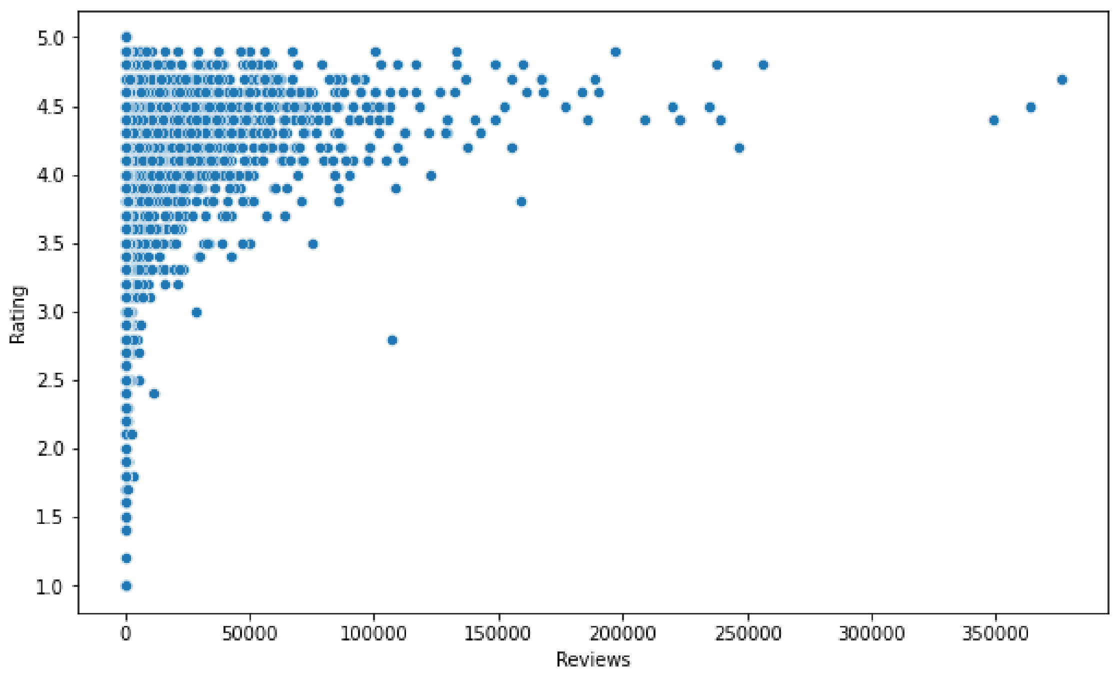


Are heavier apps rated better?

No there is no affect of size on rating but we can say heavier apps have less number of ratings.

```
[23]: # Make scatter plot/joinplot for Rating vs. Reviews
      # Does more review mean a better rating always?
      plt.figure(figsize=(10,6))
      sns.scatterplot(x='Reviews', y='Rating', data=df)
```

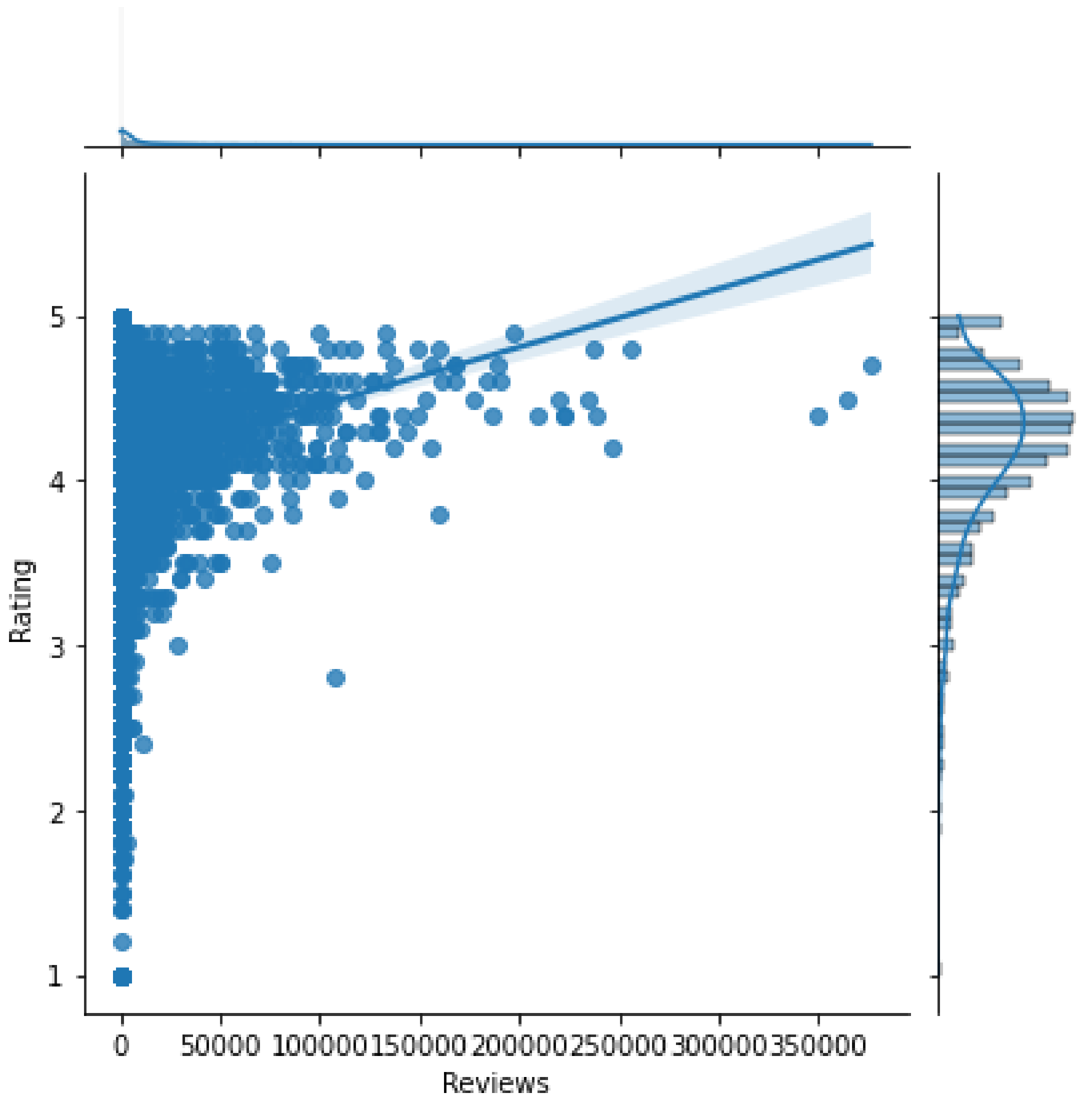
```
[23]: <AxesSubplot:xlabel='Reviews', ylabel='Rating'>
```



```
[24]: plt.figure(figsize=(10,6))
      sns.jointplot(x='Reviews', y='Rating', data=df, kind='reg')
```

```
[24]: <seaborn.axisgrid.JointGrid at 0x21fec0c6a90>

      <Figure size 720x432 with 0 Axes>
```



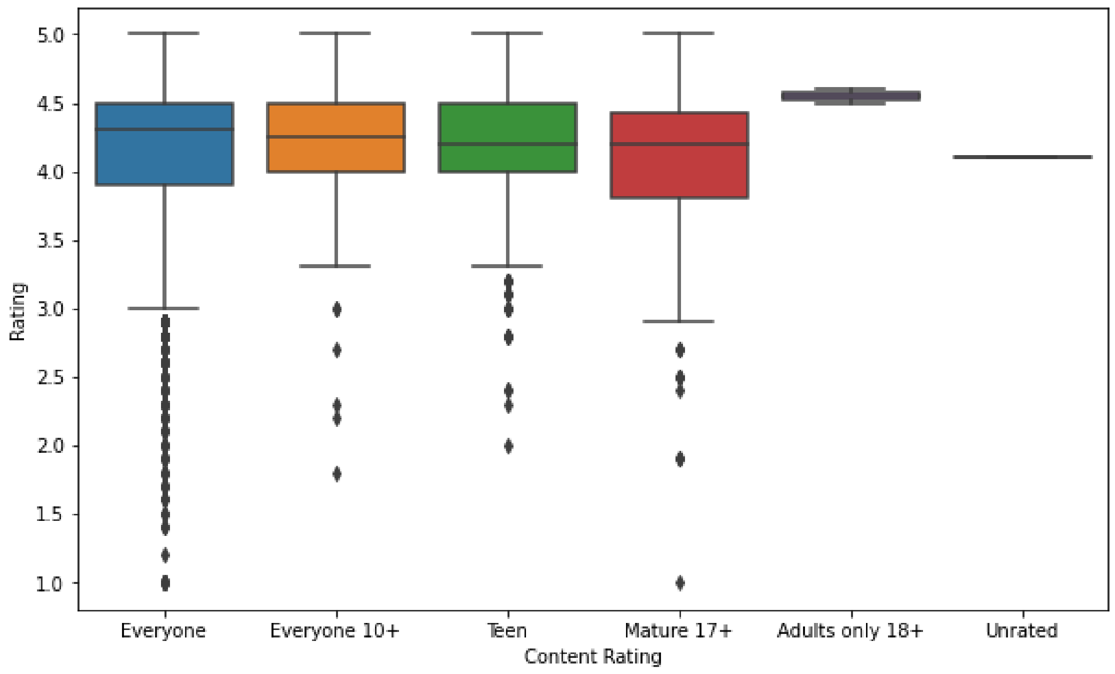
Does more review mean a better rating always?

No we can't say more reviews means better rating always. Because we can see that app with approx. 120000 reviews has las than 3 rating.

```
[25]: # 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=df)
```

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



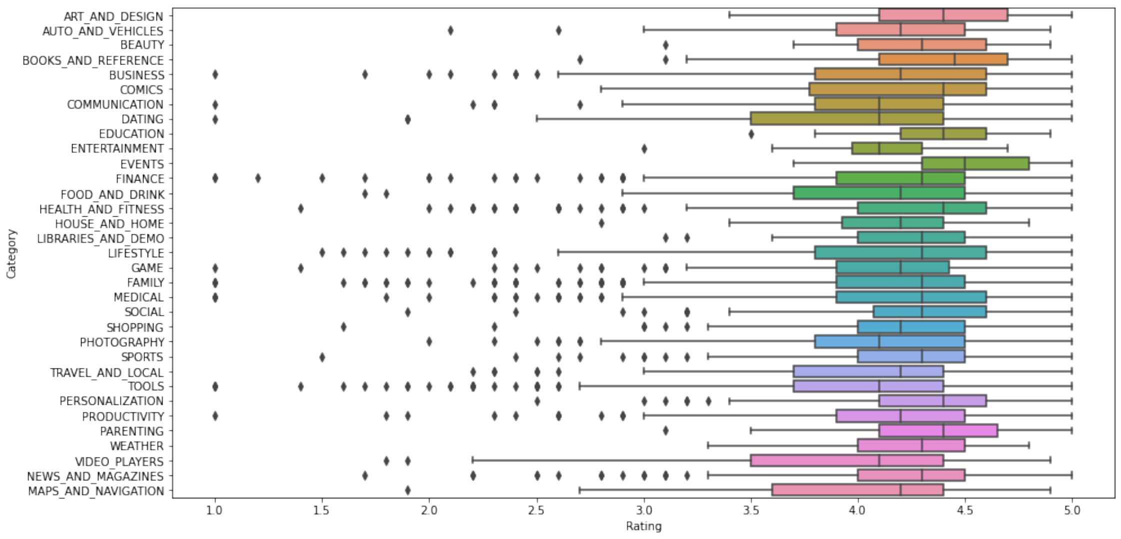
Is there any difference in the ratings? Are some types liked better?

We can say there is no much difference in average rating but definitely we can say that “Everyine 10+” and “Teen” has been liked better. Because thier most of the rating lie between 4.0 and 4.5.

[26]:
```
# Make boxplot for Ratings vs. Category
# Which genre has the best ratings?

plt.figure(figsize=(15, 8))
sns.boxplot(x='Rating', y='Category', data=df)
```

[26]:
```
<AxesSubplot:xlabel='Rating', ylabel='Category'>
```



Which genre has the best ratings?

Entertainment has the best ratings.

Data preprocessing:

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

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.

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

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.

[27]:
```
inp1 = df.copy()
```

[28]:
```
inp1.Reviews = inp1.Reviews.transform(np.log1p)
```

[29]:
```
inp1.Installs = inp1.Installs.transform(np.log1p)
```

[30]:
```
inp1.drop(['App', 'Last Updated', 'Current Ver', 'Android Ver'], axis=1,␣
 ↪inplace=True)
```

[31]:
```
cols = ['Type','Category', 'Genres', 'Content Rating']
inp2 = pd.get_dummies(inp1, columns=cols, drop_first= True)
```

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

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

[32]:
```
from sklearn.model_selection import train_test_split

df_train, df_test = train_test_split(inp2,train_size=0.7, test_size=0.3,␣
 ↪random_state=101)

y_train = df_train.pop('Rating')
X_train = df_train

y_test = df_test.pop('Rating')
X_test = df_test
```

Model building:

Use linear regression as the technique

Report the R2 on the train set

[33]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

lr = LinearRegression()

lr.fit(X_train, y_train)

y_train_pred = lr.predict(X_train)

r2_score(y_train, y_train_pred)
```

[33]:
```
0.15918782652625463
```

Make predictions on test set and report R2.

[34]:
```
y_test_pred = lr.predict(X_test)
r2_score(y_test, y_test_pred)
```

[34]:
```
0.0868353801993379
```

[ ]:
```

```