appratingprediction

September 23, 2023

1 App Rating Prediction

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:

Load the data file using pandas.

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

Drop records with nulls in any of the columns.

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. 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.

5. 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?

6. 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?

Drop these as most seem to be junk apps

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

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.

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.

8. 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.

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.

11 . Model building

Use linear regression as the technique

Report the R2 on the train set

12. Make predictions on test set and report R2.

Import libraries

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import skew
import warnings
warnings.filterwarnings('ignore')
```

1.1 Load the Data

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

Data Description

[3]:
```
df.head()
```

[3]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current 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
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
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
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	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design	June 20, 2018	1.1

```
4    Art & Design;Creativity     June 20, 2018                1.1

    Android Ver
0  4.0.3 and up
1  4.0.3 and up
2  4.0.3 and up
3    4.2 and up
4    4.4 and up
```

[4]:
```
df.describe()
```

[4]:
```
             Rating
count  9367.000000
mean      4.193338
std       0.537431
min       1.000000
25%       4.000000
50%       4.300000
75%       4.500000
max      19.000000
```

[5]:
```
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
```

Check Null Values

[6]:
```
df.isnull()
```

[6]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
…	…	…	…	…	…	…	…	…
10836	False	False	False	False	False	False	False	False
10837	False	False	False	False	False	False	False	False
10838	False	False	True	False	False	False	False	False
10839	False	False	False	False	False	False	False	False
10840	False	False	False	False	False	False	False	False

	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
…	…	…	…	…	…
10836	False	False	False	False	False
10837	False	False	False	False	False
10838	False	False	False	False	False
10839	False	False	False	False	False
10840	False	False	False	False	False

[10841 rows x 13 columns]

Null values count by each column.

[7]:
```
df.isnull().sum()
```

[7]:
```
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
```

Data Wrangling Droping the records with null in any of the column

[8]:
```
df.dropna(inplace= True)
```

[9]:
```
df.isnull().sum()
```

[9]:
```
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
```

Checking the revised Rows and columns

[10]:
```
df.reset_index(drop= True, inplace = True)
df.shape
```

[10]:
```
(9360, 13)
```

[11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9360 entries, 0 to 9359
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   App             9360 non-null   object
 1   Category        9360 non-null   object
 2   Rating          9360 non-null   float64
 3   Reviews         9360 non-null   object
 4   Size            9360 non-null   object
 5   Installs        9360 non-null   object
 6   Type            9360 non-null   object
 7   Price           9360 non-null   object
 8   Content Rating  9360 non-null   object
 9   Genres          9360 non-null   object
 10  Last Updated    9360 non-null   object
 11  Current Ver     9360 non-null   object
 12  Android Ver     9360 non-null   object
```

```
dtypes: float64(1), object(12)
memory usage: 950.8+ KB
```

[12]:
```
df['Size'].unique()
```

[12]:
```
array(['19M', '14M', '8.7M', '25M', '2.8M', '5.6M', '29M', '33M', '3.1M',
       '28M', '12M', '20M', '21M', '37M', '5.5M', '17M', '39M', '31M',
       '4.2M', '23M', '6.0M', '6.1M', '4.6M', '9.2M', '5.2M', '11M',
       '24M', 'Varies with device', '9.4M', '15M', '10M', '1.2M', '26M',
       '8.0M', '7.9M', '56M', '57M', '35M', '54M', '201k', '3.6M', '5.7M',
       '8.6M', '2.4M', '27M', '2.7M', '2.5M', '7.0M', '16M', '3.4M',
       '8.9M', '3.9M', '2.9M', '38M', '32M', '5.4M', '18M', '1.1M',
       '2.2M', '4.5M', '9.8M', '52M', '9.0M', '6.7M', '30M', '2.6M',
       '7.1M', '22M', '6.4M', '3.2M', '8.2M', '4.9M', '9.5M', '5.0M',
       '5.9M', '13M', '73M', '6.8M', '3.5M', '4.0M', '2.3M', '2.1M',
       '42M', '9.1M', '55M', '23k', '7.3M', '6.5M', '1.5M', '7.5M', '51M',
       '41M', '48M', '8.5M', '46M', '8.3M', '4.3M', '4.7M', '3.3M', '40M',
       '7.8M', '8.8M', '6.6M', '5.1M', '61M', '66M', '79k', '8.4M',
       '3.7M', '118k', '44M', '695k', '1.6M', '6.2M', '53M', '1.4M',
       '3.0M', '7.2M', '5.8M', '3.8M', '9.6M', '45M', '63M', '49M', '77M',
       '4.4M', '70M', '9.3M', '8.1M', '36M', '6.9M', '7.4M', '84M', '97M',
       '2.0M', '1.9M', '1.8M', '5.3M', '47M', '556k', '526k', '76M',
       '7.6M', '59M', '9.7M', '78M', '72M', '43M', '7.7M', '6.3M', '334k',
       '93M', '65M', '79M', '100M', '58M', '50M', '68M', '64M', '34M',
       '67M', '60M', '94M', '9.9M', '232k', '99M', '624k', '95M', '8.5k',
       '41k', '292k', '80M', '1.7M', '10.0M', '74M', '62M', '69M', '75M',
       '98M', '85M', '82M', '96M', '87M', '71M', '86M', '91M', '81M',
       '92M', '83M', '88M', '704k', '862k', '899k', '378k', '4.8M',
       '266k', '375k', '1.3M', '975k', '980k', '4.1M', '89M', '696k',
       '544k', '525k', '920k', '779k', '853k', '720k', '713k', '772k',
       '318k', '58k', '241k', '196k', '857k', '51k', '953k', '865k',
       '251k', '930k', '540k', '313k', '746k', '203k', '26k', '314k',
       '239k', '371k', '220k', '730k', '756k', '91k', '293k', '17k',
       '74k', '14k', '317k', '78k', '924k', '818k', '81k', '939k', '169k',
       '45k', '965k', '90M', '545k', '61k', '283k', '655k', '714k', '93k',
       '872k', '121k', '322k', '976k', '206k', '954k', '444k', '717k',
       '210k', '609k', '308k', '306k', '175k', '350k', '383k', '454k',
       '1.0M', '70k', '812k', '442k', '842k', '417k', '412k', '459k',
       '478k', '335k', '782k', '721k', '430k', '429k', '192k', '460k',
       '728k', '496k', '816k', '414k', '506k', '887k', '613k', '778k',
       '683k', '592k', '186k', '840k', '647k', '373k', '437k', '598k',
       '716k', '585k', '982k', '219k', '55k', '323k', '691k', '511k',
       '951k', '963k', '25k', '554k', '351k', '27k', '82k', '208k',
       '551k', '29k', '103k', '116k', '153k', '209k', '499k', '173k',
       '597k', '809k', '122k', '411k', '400k', '801k', '787k', '50k',
       '643k', '986k', '516k', '837k', '780k', '20k', '498k', '600k',
       '656k', '221k', '228k', '176k', '34k', '259k', '164k', '458k',
```

```
       '629k', '28k', '288k', '775k', '785k', '636k', '916k', '994k',
       '309k', '485k', '914k', '903k', '608k', '500k', '54k', '562k',
       '847k', '948k', '811k', '270k', '48k', '523k', '784k', '280k',
       '24k', '892k', '154k', '18k', '33k', '860k', '364k', '387k',
       '626k', '161k', '879k', '39k', '170k', '141k', '160k', '144k',
       '143k', '190k', '376k', '193k', '473k', '246k', '73k', '253k',
       '957k', '420k', '72k', '404k', '470k', '226k', '240k', '89k',
       '234k', '257k', '861k', '467k', '676k', '552k', '582k', '619k'],
      dtype=object)
```

Convert all categorical data types into numeric

Start the cleaning with Size Column and converting in to numeric

[13]:
```
df['Size'] = df['Size'].apply(lambda x: str(x).replace('M', '') if 'M' in␣
 ↪str(x) else x)
df['Size'] = df['Size'].apply(lambda x: str(x).replace('Varies with device',␣
 ↪'nan') if 'Varies with device' in str(x) else x)

# Scaling all the values to Millions format (means that 19.0 => 19x10^6 => 19M)
df['Size'] = df['Size'].apply(lambda x: float(str(x).replace('k', ''))/1000 if␣
 ↪'k' in str(x) else x)
df['Size'] = df['Size'].apply(lambda x : float(x))
df = df[pd.notnull(df['Size'])]
df['Size'].dtype
```

[13]:
```
dtype('float64')
```

[14]:
```
df.shape
```

[14]:
```
(7723, 13)
```

[15]:
```
df['Reviews'].unique()
```

[15]:
```
array(['159', '967', '87510', …, '603', '1195', '398307'], dtype=object)
```

Converting the Reviews column

[16]:
```
df['Reviews'] = df['Reviews'].apply(lambda x : int(x))
df['Reviews'].dtype
```

[16]:
```
dtype('int64')
```

[17]:
```
df['Rating'].dtype
```

[17]:
```
dtype('float64')
```

df[‘Price’].unique()

[18]:
```
df['Price'] = df['Price'].apply((lambda x:str(x).replace('$','') if '$' in␣
 ↪str(x) else str(x)))
df['Price'] = df['Price'].apply (lambda x: float(x))
df['Price'].dtype
```

[18]:
```
dtype('float64')
```

[19]:
```
df.shape
```

[19]:
```
(7723, 13)
```

[20]:
```
df['Installs'].unique()
```

[20]:
```
array(['10,000+', '500,000+', '5,000,000+', '50,000,000+', '100,000+',
       '50,000+', '1,000,000+', '10,000,000+', '5,000+', '100,000,000+',
       '1,000+', '500,000,000+', '100+', '500+', '10+', '1,000,000,000+',
       '5+', '50+', '1+'], dtype=object)
```

Cleaning and conversion of the Installs column

[21]:
```
df['Installs'] = df['Installs'].apply (lambda x: str(x).replace('+','') if '+'␣
 ↪in str(x) else x)
df['Installs'] = df['Installs'].apply(lambda x: str(x).replace(',', '') if ','␣
 ↪in str(x) else x)
df['Installs'] = df['Installs'].apply(lambda x: int (x))
df['Installs'].dtype
```

[21]:
```
dtype('int64')
```

[22]:
```
df.shape
```

[22]:
```
(7723, 13)
```

1.1.1 Sanity Check

The Play Store only accepts ratings within the range of 1 to 5.

[23]:
```
df['Rating'].unique()
```

[23]:
```
array([4.1, 3.9, 4.7, 4.5, 4.3, 4.4, 3.8, 4.2, 4.6, 4. , 4.8, 4.9, 3.6,
       3.7, 3.2, 3.3, 3.4, 3.5, 3.1, 5. , 2.6, 3. , 1.9, 2.5, 2.8, 2.7,
       1. , 2.9, 2.3, 2.2, 1.7, 2. , 1.8, 2.4, 1.6, 2.1, 1.4, 1.5, 1.2])
```

[24]:
```
df[df['Rating']>5].shape[0]
```

[24]:
```
0
```

[25]:
```
df[df['Reviews']>df['Installs']].shape[0]
```

[25]:
```
6
```

Found 6 of reviews that were more than the Installs. let's Drop to get the unbiased Dataset

[26]:
```
df.shape
```

[26]:
```
(7723, 13)
```

[27]:
```
df.drop(df[df['Reviews']>df['Installs']].index, inplace = True)
```

For free apps price should be equal to 0

[28]:
```
df[(df['Type']=='free')& (df['Price'] ==0)].shape[0]
```

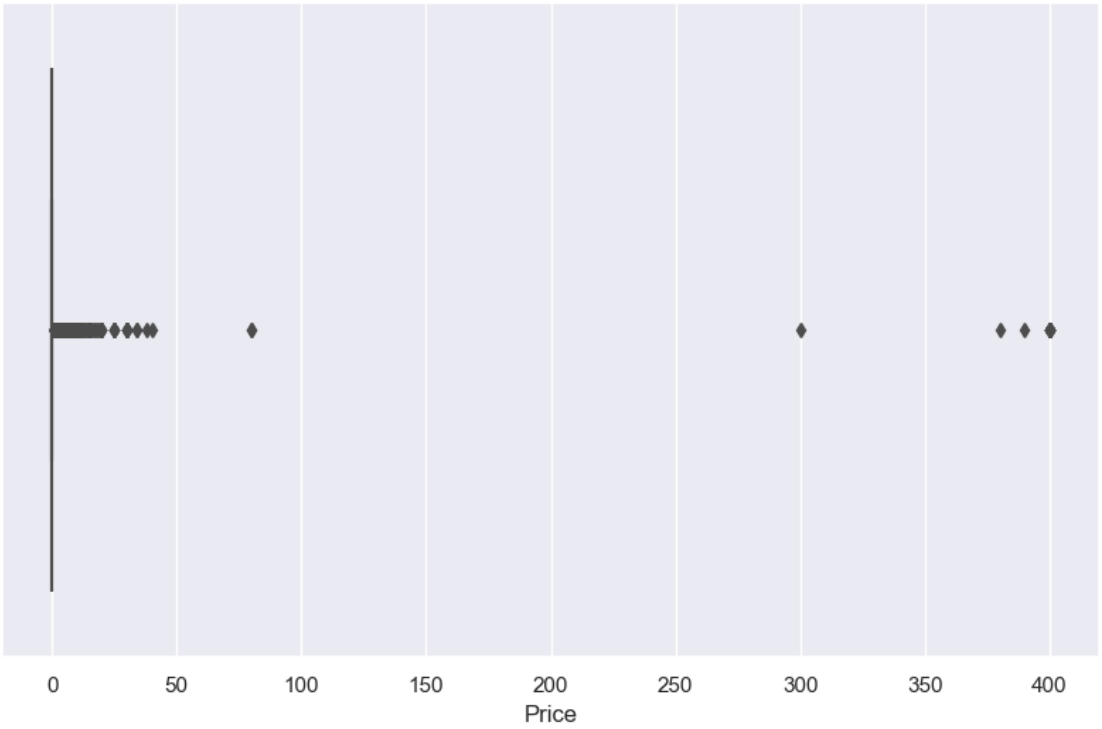
[28]:
```
0
```

1.2 Univariate Analysis

Box Plot for Price

[29]:
```
sns.set(rc={'figure.figsize':(10,6)})
```

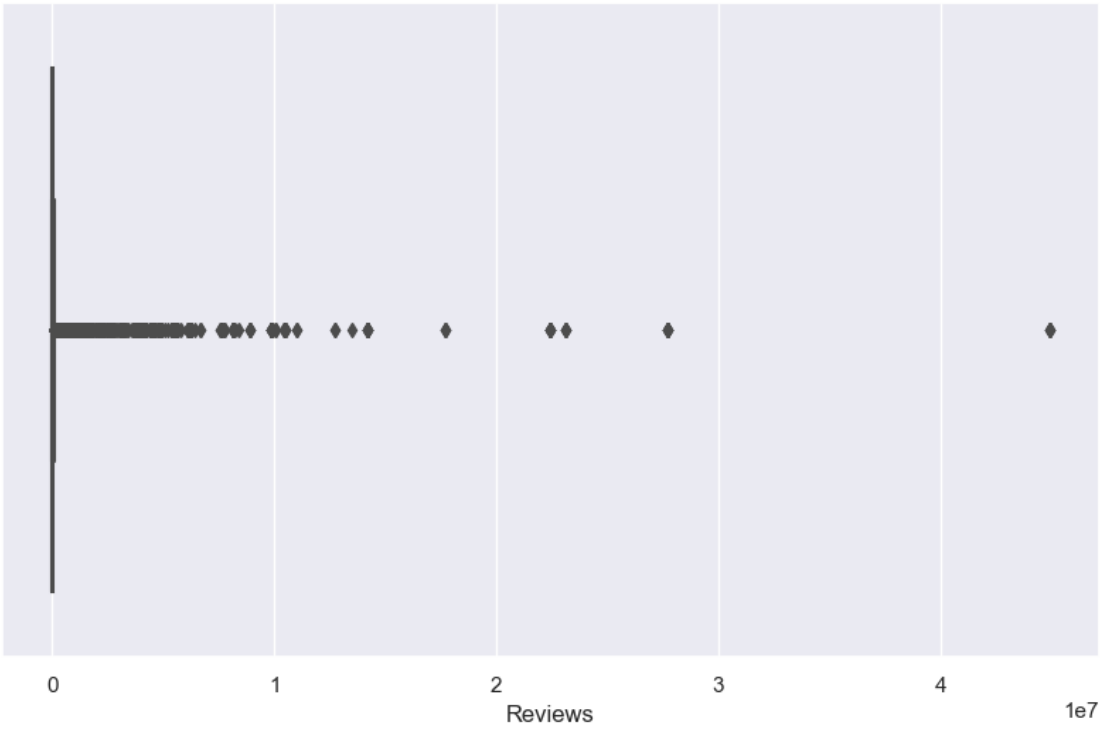
[30]:
```
sns.boxplot(x= 'Price',data= df);
```



Certainly, there are a few outliers in the Price column, indicating that some apps on the Google Play Store have prices significantly higher than the typical ones.

Boxplot for Reviews

[31]:
```
sns.boxplot(x ='Reviews', data =df);
```

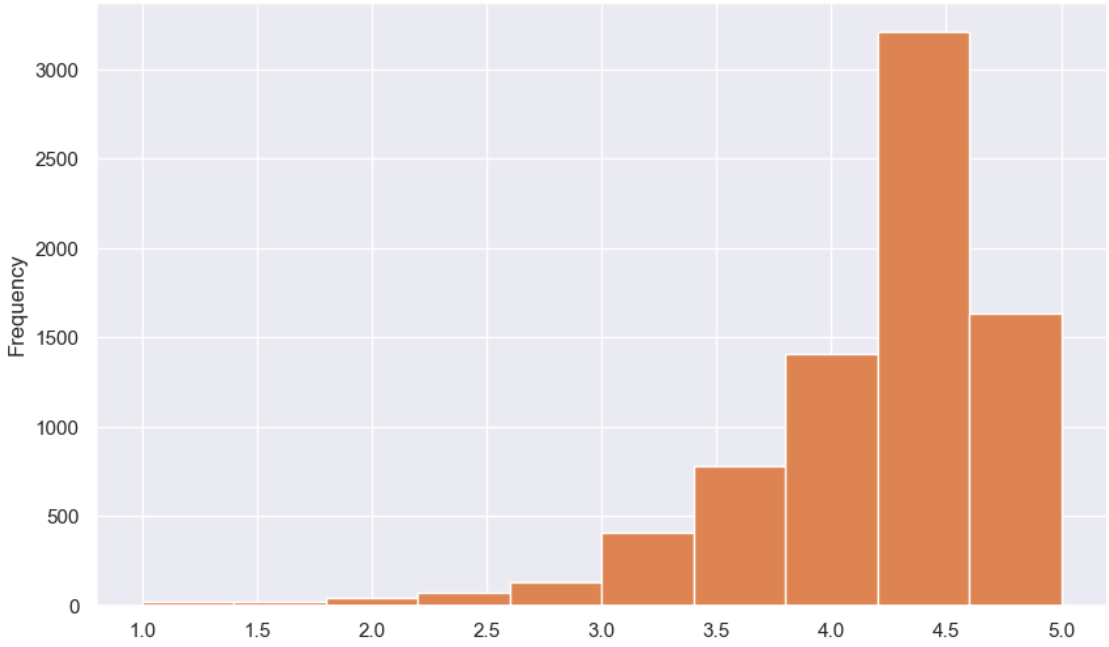


Indeed, there are certain apps with an exceptionally high number of reviews on the Google Play Store.

Histogram for Rating

[32]:
```
df['Rating'].plot(kind= 'hist'); #we can use either to get the results
plt.hist(df['Rating'])
```

[32]:
```
(array([  17.,   18.,   39.,   72.,  132.,  408.,  781., 1406., 3212.,
        1632.]),
 array([1. , 1.4, 1.8, 2.2, 2.6, 3. , 3.4, 3.8, 4.2, 4.6, 5. ]),
 <BarContainer object of 10 artists>)
```



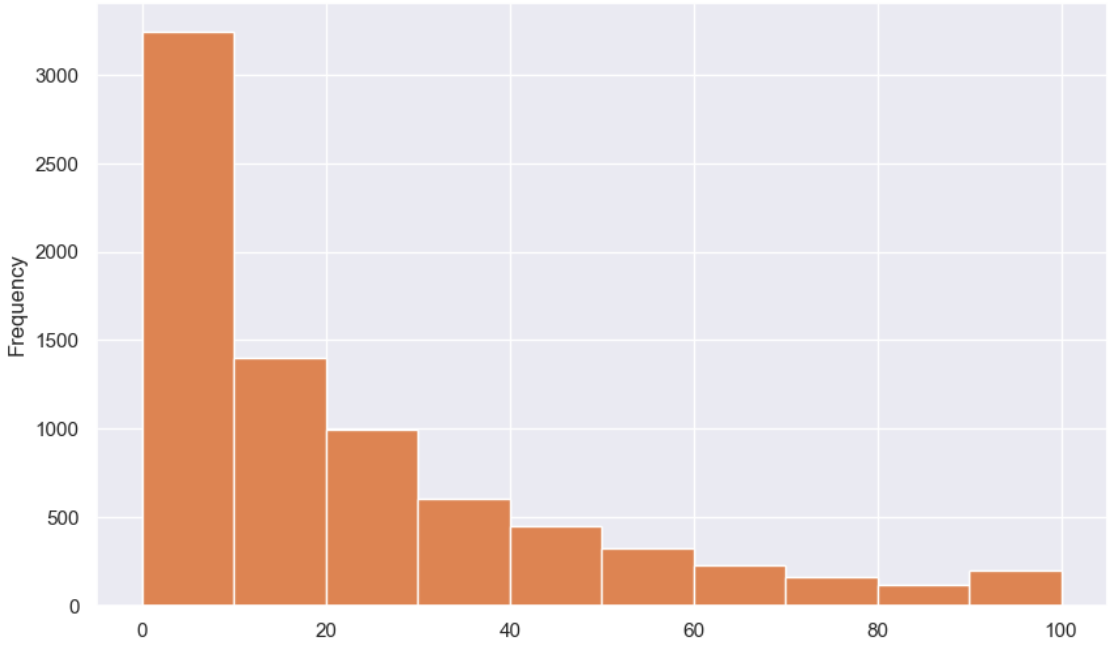
There is a Negative skewness(left- skewed)

some apps seem to have higher Ratings than usual

Histogram for Size

[33]:
```
df['Size'].plot(kind= 'hist') #we can use either to get the results
plt.hist(df['Size'])
```

[33]:
```
(array([3245., 1398.,  991.,  606.,  449.,  325.,  226.,  161.,  117.,
         199.]),
 array([8.500000e-03, 1.000765e+01, 2.000680e+01, 3.000595e+01,
        4.000510e+01, 5.000425e+01, 6.000340e+01, 7.000255e+01,
        8.000170e+01, 9.000085e+01, 1.000000e+02]),
 <BarContainer object of 10 artists>)
```



positive skewness Right Skewed

1.3 Handling Outliers

As per the above observation of plots, there seems to be some outliers in the Price & Reviews column

In the Installs column as well , price of $200 and above for an application is expected to be very high

[34]:
```
df[df['Price']>200].index.shape[0] #we can use either to get the results
df.loc[df['Price']>200].shape[0]
```

[34]: 15

Dropping the Junk apps

[35]:
```
df.drop(df[df['Price']>200].index, inplace= True)
```

[36]:
```
df.shape
```

[36]: (7702, 13)

A limited number of apps boast an exceptionally high number of reviews on the Google Play Store.

[37]:
```
df.loc[df['Reviews']>2000000].shape[0]
```

[37]: 219

After excluding the apps with a high number of stars, it's important to assess the data's shape to ensure that the analysis remains representative.

[38]:
```
df.drop(df[df['Reviews']>2000000].index, inplace= True)
df.shape
```

[38]: (7483, 13)

Find out the Percentiles of Installs and decide a threshold as cutoff for outlier dropping the value more than the cutoff(threshold -95th percentile)

[39]:
```
df.drop(df[df['Installs']>10000000].index, inplace= True)
```

[40]:
```
df.shape
```

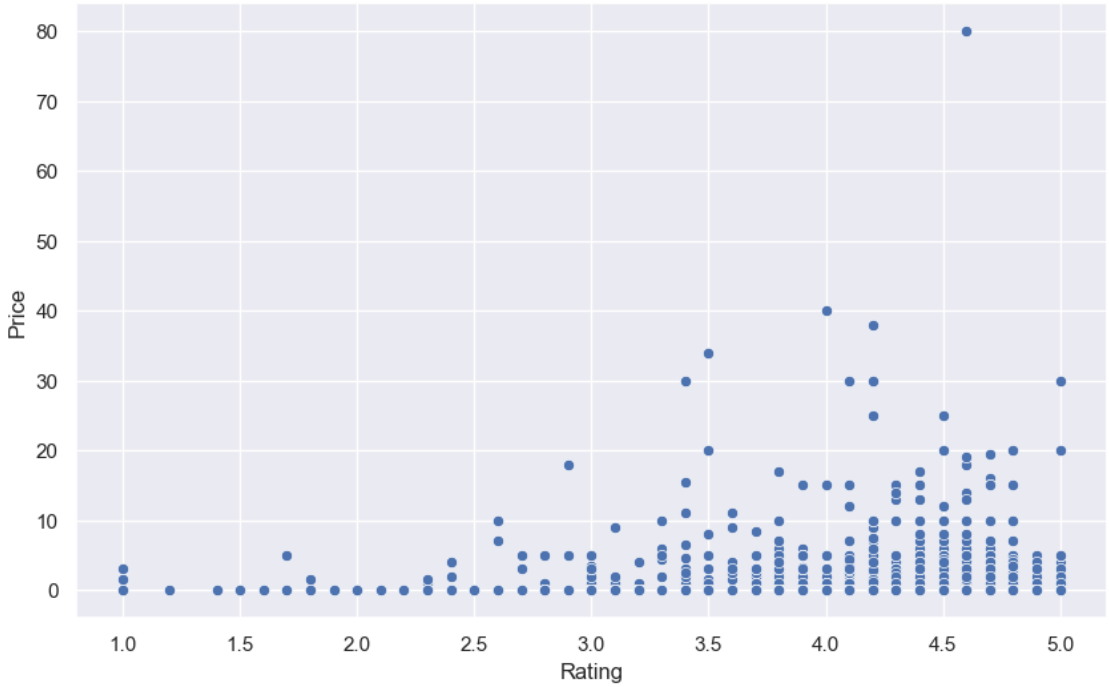
[40]: (7307, 13)

1.4 Bivariate analysis

Scatter plot/jointplot for Rating Vs. Price

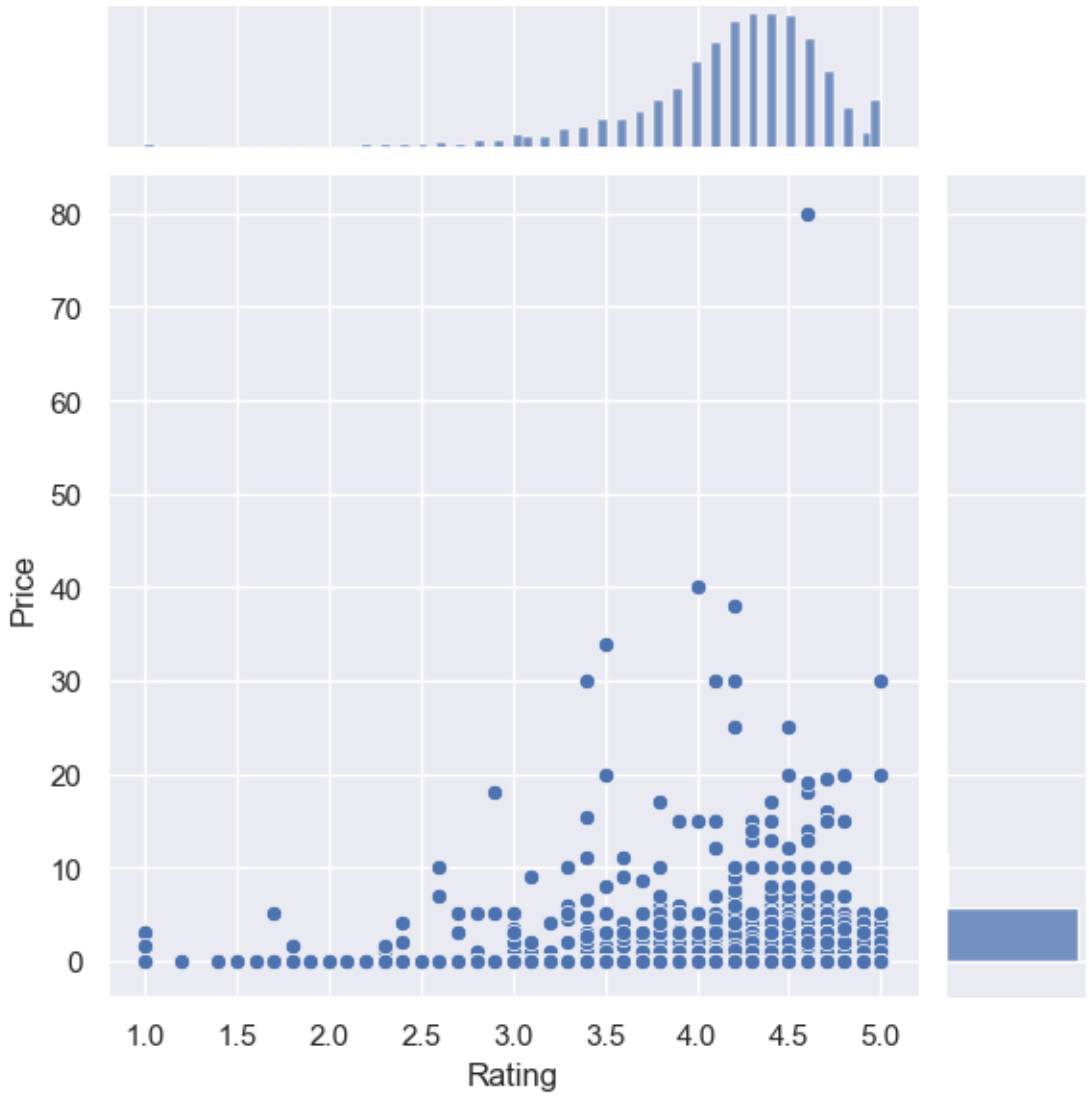
[41]:
```
sns.scatterplot(x = 'Rating', y = 'Price',data=df)
```

[41]: <Axes: xlabel='Rating', ylabel='Price'>



[42]:
```
sns.jointplot(x= 'Rating',y= 'Price',data= df)
```

[42]:
```
<seaborn.axisgrid.JointGrid at 0x1612fc6f290>
```

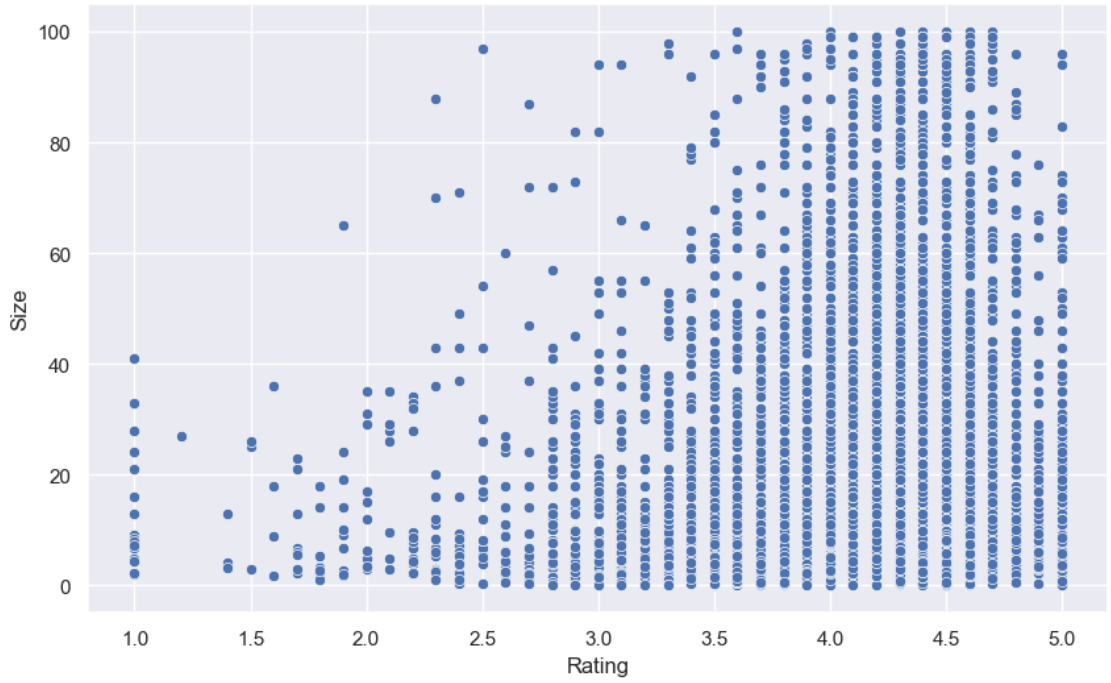


Both plots depict a positive linear relationship, suggesting that as the price of an app increases, its rating tends to increase as well. This observation implies that paid apps often have higher ratings compared to free ones.

Scatterplot/jointplot for Rating Vs. Size

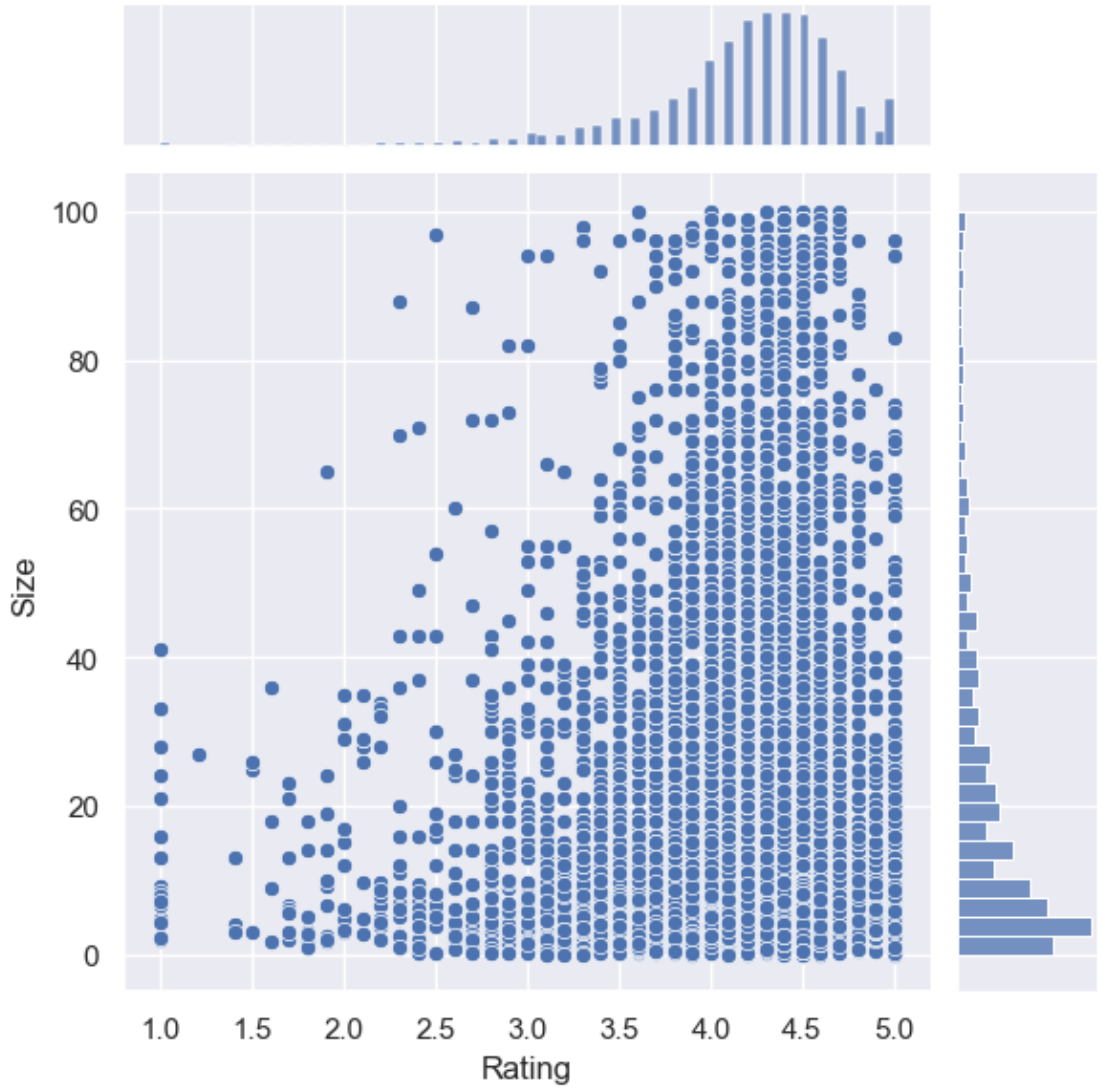
[43]:
```
sns.scatterplot(x= 'Rating',y= 'Size', data= df)
```

[43]: <Axes: xlabel='Rating', ylabel='Size'>



[44]:
```
sns.jointplot(x= 'Rating', y= 'Size', data= df)
```

[44]: <seaborn.axisgrid.JointGrid at 0x161320b3ad0>

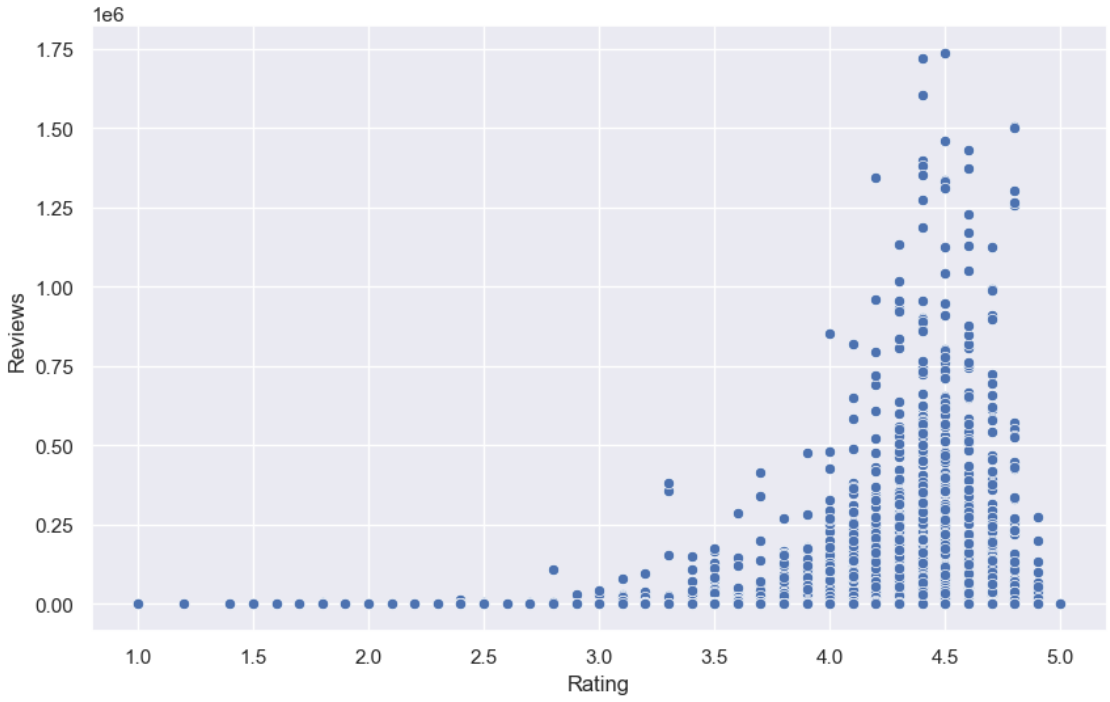


The plots demonstrate a positive linear relationship, indicating that as the size of an app increases, its ratings also tend to increase. This suggests that larger apps are generally rated more favorably by users.

Scatterplot for Ratings Vs. Reviews

[45]:
```
sns.scatterplot(x= 'Rating',y= 'Reviews', data= df)
```

[45]:
```
<Axes: xlabel='Rating', ylabel='Reviews'>
```

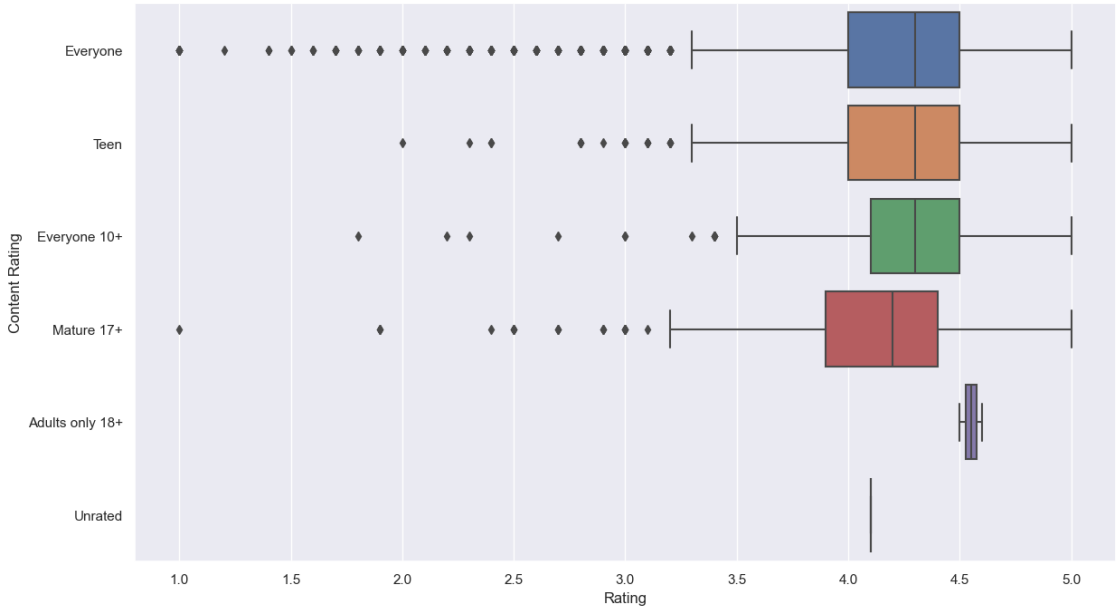


The plot shows a positive linear relationship between Ratings and Reviews. More reviews mean better ratings indeed

Boxplot for Ratings Vs. Content Rating

[46]:
```
sns.set(rc={'figure.figsize':(14,8)})
sns.boxplot(x= 'Rating', y= 'Content Rating', data = df)
```

[46]:
```
<Axes: xlabel='Rating', ylabel='Content Rating'>
```



Based on the plot and the presence of outliers, it appears that apps rated for "Everyone" have the lowest average ratings and the highest number of outliers, suggesting that they are generally rated lower. The categories "Mature 17+" and "Everyone 10+" also exhibit a similar trend with a notable number of outliers, indicating lower ratings on average.

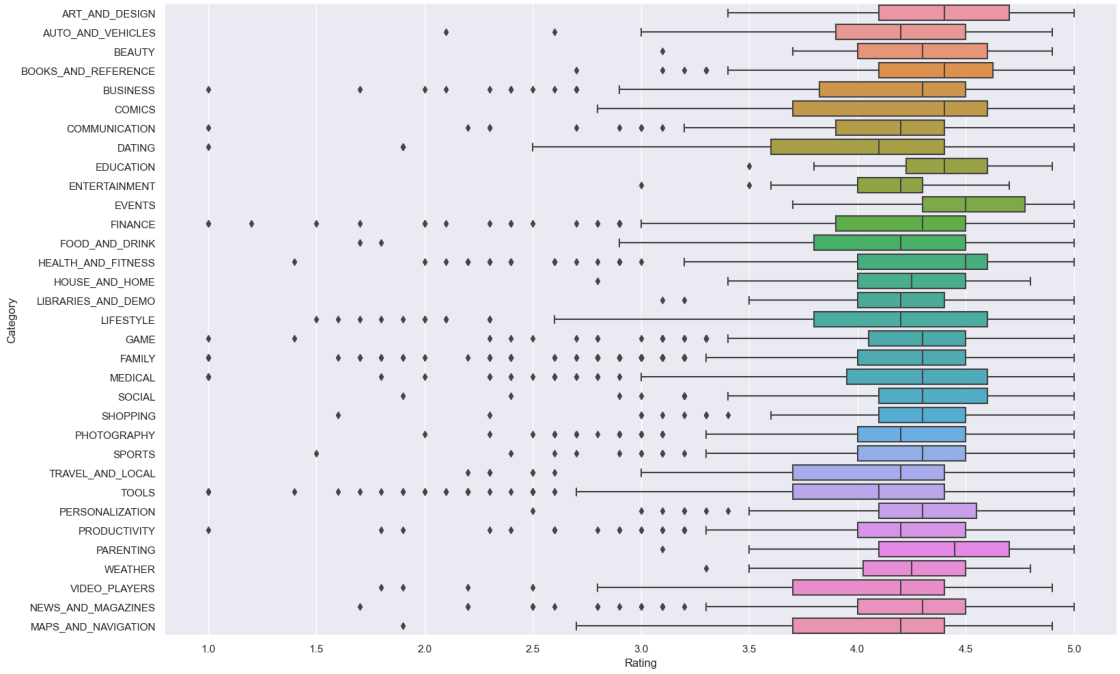
In contrast, the "Adults only 18+" category appears to have better ratings on average and fewer outliers, suggesting that apps in this category tend to be better received and liked by users.

It's important to note that while these observations can be made based on the plot, further statistical analysis and domain-specific context may be necessary to draw definitive conclusions about app ratings across different age categories.

Boxplot for Ratings Vs. Category

[47]:
```
sns.set(rc={'figure.figsize':(18,12)})
sns.boxplot(x= 'Rating', y = 'Category', data= df)
```

[47]:
```
<Axes: xlabel='Rating', ylabel='Category'>
```



From the above plot the Category Events has the best Ratings out of all other app genres

1.5 Data Preprocessing

1.6 Model development

creating a copy of the data(df) to make all edits

[48]:
```
inp1= df.copy()
```

[49]:
```
inp1.head()
```

[49]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19.0	10000	Free	0.0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14.0	500000	Free	0.0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide …	ART_AND_DESIGN	4.7	87510	8.7	5000000	Free	0.0	Everyone
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8	100000	Free	0.0	Everyone
5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5.6	50000	Free	0.0	Everyone

```
                      Genres      Last Updated Current Ver   Android Ver
0               Art & Design   January 7, 2018       1.0.0  4.0.3 and up
1  Art & Design;Pretend Play  January 15, 2018       2.0.0  4.0.3 and up
2               Art & Design    August 1, 2018       1.2.4  4.0.3 and up
4    Art & Design;Creativity     June 20, 2018         1.1    4.4 and up
5               Art & Design    March 26, 2017         1.0    2.3 and up
```

Reviews and Installs column still have some relatively high values, before building the linear regression model we need to reduce the skew; columns needs log transformation

apply log transformation to Reviews

[50]:
```
reviews_skew = np.log1p(inp1['Reviews'])
inp1['Reviews']= reviews_skew
```

[51]:
```
reviews_skew.skew()
```

[51]:
```
-0.06808430177422442
```

apply log transformation to Installs

[52]:
```
Installs_skew = np.log1p(inp1['Installs'])
inp1['Installs']
```

[52]:
```
0          10000
1         500000
2        5000000
4         100000
5          50000
          ...
9354        1000
9355         500
9356        5000
9357         100
9359    10000000
Name: Installs, Length: 7307, dtype: int64
```

[53]:
```
Installs_skew.skew()
```

[53]:
```
-0.3930918801065247
```

[54]:
```
inp1.head()
```

[54]:
```
                                                 App        Category  Rating  \
0     Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN     4.1
1                                Coloring book moana  ART_AND_DESIGN     3.9
2  U Launcher Lite - FREE Live Cool Themes, Hide ...  ART_AND_DESIGN     4.7
4              Pixel Draw - Number Art Coloring Book  ART_AND_DESIGN     4.3
```

```
5                        Paper flowers instructions  ART_AND_DESIGN     4.4

     Reviews  Size  Installs  Type  Price Content Rating  \
0   5.075174  19.0     10000  Free    0.0       Everyone
1   6.875232  14.0    500000  Free    0.0       Everyone
2  11.379520   8.7   5000000  Free    0.0       Everyone
4   6.875232   2.8    100000  Free    0.0       Everyone
5   5.123964   5.6     50000  Free    0.0       Everyone

                       Genres      Last Updated Current Ver   Android Ver
0               Art & Design   January 7, 2018       1.0.0  4.0.3 and up
1  Art & Design;Pretend Play  January 15, 2018       2.0.0  4.0.3 and up
2               Art & Design    August 1, 2018       1.2.4  4.0.3 and up
4    Art & Design;Creativity     June 20, 2018         1.1    4.4 and up
5               Art & Design    March 26, 2017         1.0    2.3 and up
```

[55]:
```
inp1.drop(['App','Last Updated','Current Ver','Android Ver','Type'], axis= 1,␣
↪inplace = True)
```

[56]:
```
inp1.head()
```

[56]:
```
          Category  Rating    Reviews  Size  Installs  Price Content Rating  \
0  ART_AND_DESIGN     4.1   5.075174  19.0     10000    0.0       Everyone
1  ART_AND_DESIGN     3.9   6.875232  14.0    500000    0.0       Everyone
2  ART_AND_DESIGN     4.7  11.379520   8.7   5000000    0.0       Everyone
4  ART_AND_DESIGN     4.3   6.875232   2.8    100000    0.0       Everyone
5  ART_AND_DESIGN     4.4   5.123964   5.6     50000    0.0       Everyone

                       Genres
0               Art & Design
1  Art & Design;Pretend Play
2               Art & Design
4    Art & Design;Creativity
5               Art & Design
```

[57]:
```
inp1.shape
```

[57]:
```
(7307, 8)
```

As Model does not understand any Catergorical variable hence these need to be converted to numerical

Dummy Encoding is one way to convert these columns into numerical

create a copy of dataframe

[58]:
```
inp2 = inp1
```

[59]:
```
inp2.head()
```

[59]:
```
          Category  Rating    Reviews  Size  Installs  Price Content Rating  \
0  ART_AND_DESIGN     4.1   5.075174  19.0     10000    0.0       Everyone
1  ART_AND_DESIGN     3.9   6.875232  14.0    500000    0.0       Everyone
2  ART_AND_DESIGN     4.7  11.379520   8.7   5000000    0.0       Everyone
4  ART_AND_DESIGN     4.3   6.875232   2.8    100000    0.0       Everyone
5  ART_AND_DESIGN     4.4   5.123964   5.6     50000    0.0       Everyone

                      Genres
0               Art & Design
1  Art & Design;Pretend Play
2               Art & Design
4    Art & Design;Creativity
5               Art & Design
```

get unique values in column category

[60]:
```
inp2['Category'].unique()
```

[60]:
```
array(['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'],
      dtype=object)
```

[61]:
```
inp2.Category = pd.Categorical(inp2.Category)

x = inp2[['Category']]
del inp2['Category']

dummies = pd.get_dummies(x, prefix = 'Category')
inp2 = pd.concat([inp2,dummies], axis=1)
inp2.head()
```

[61]:
```
   Rating    Reviews  Size  Installs  Price Content Rating  \
0     4.1   5.075174  19.0     10000    0.0       Everyone
1     3.9   6.875232  14.0    500000    0.0       Everyone
2     4.7  11.379520   8.7   5000000    0.0       Everyone
4     4.3   6.875232   2.8    100000    0.0       Everyone
5     4.4   5.123964   5.6     50000    0.0       Everyone

                      Genres  Category_ART_AND_DESIGN  \
```

```
0              Art & Design                     True
1  Art & Design;Pretend Play                    True
2              Art & Design                     True
4   Art & Design;Creativity                     True
5              Art & Design                     True

   Category_AUTO_AND_VEHICLES  Category_BEAUTY  …  Category_PERSONALIZATION  \
0                       False            False  …                     False
1                       False            False  …                     False
2                       False            False  …                     False
4                       False            False  …                     False
5                       False            False  …                     False

   Category_PHOTOGRAPHY  Category_PRODUCTIVITY  Category_SHOPPING  \
0                 False                  False              False
1                 False                  False              False
2                 False                  False              False
4                 False                  False              False
5                 False                  False              False

   Category_SOCIAL  Category_SPORTS  Category_TOOLS  \
0            False            False           False
1            False            False           False
2            False            False           False
4            False            False           False
5            False            False           False

   Category_TRAVEL_AND_LOCAL  Category_VIDEO_PLAYERS  Category_WEATHER
0                      False                   False             False
1                      False                   False             False
2                      False                   False             False
4                      False                   False             False
5                      False                   False             False

[5 rows x 40 columns]
```

get unique values in Column Genres

[62]:
```
inp2["Genres"].unique()
```

[62]:
```
array(['Art & Design', 'Art & Design;Pretend Play',
       'Art & Design;Creativity', 'Auto & Vehicles', 'Beauty',
       'Books & Reference', 'Business', 'Comics', 'Comics;Creativity',
       'Communication', 'Dating', 'Education', 'Education;Creativity',
       'Education;Education', 'Education;Action & Adventure',
       'Education;Pretend Play', 'Education;Brain Games', 'Entertainment',
       'Entertainment;Brain Games', 'Entertainment;Music & Video',
```

```
'Events', 'Finance', 'Food & Drink', 'Health & Fitness',
'House & Home', 'Libraries & Demo', 'Lifestyle',
'Lifestyle;Pretend Play', 'Card', 'Casual', 'Puzzle', 'Action',
'Arcade', 'Word', 'Racing', 'Casual;Creativity', 'Sports', 'Board',
'Simulation', 'Role Playing', 'Strategy', 'Simulation;Education',
'Action;Action & Adventure', 'Trivia', 'Casual;Brain Games',
'Simulation;Action & Adventure', 'Educational;Creativity',
'Puzzle;Brain Games', 'Educational;Education', 'Card;Brain Games',
'Educational;Brain Games', 'Educational;Pretend Play',
'Casual;Action & Adventure', 'Entertainment;Education',
'Casual;Education', 'Casual;Pretend Play', 'Music;Music & Video',
'Arcade;Pretend Play', 'Adventure;Action & Adventure',
'Simulation;Pretend Play', 'Puzzle;Creativity',
'Racing;Action & Adventure', 'Educational;Action & Adventure',
'Arcade;Action & Adventure', 'Entertainment;Action & Adventure',
'Puzzle;Action & Adventure', 'Role Playing;Action & Adventure',
'Strategy;Action & Adventure', 'Music & Audio;Music & Video',
'Health & Fitness;Education', 'Adventure;Education',
'Board;Brain Games', 'Board;Action & Adventure',
'Board;Pretend Play', 'Casual;Music & Video',
'Education;Music & Video', 'Role Playing;Pretend Play',
'Entertainment;Pretend Play', 'Video Players & Editors;Creativity',
'Card;Action & Adventure', 'Medical', 'Social', 'Shopping',
'Photography', 'Travel & Local',
'Travel & Local;Action & Adventure', 'Tools', 'Personalization',
'Productivity', 'Parenting', 'Parenting;Brain Games',
'Parenting;Education', 'Parenting;Music & Video', 'Weather',
'Video Players & Editors', 'News & Magazines', 'Maps & Navigation',
'Adventure', 'Health & Fitness;Action & Adventure', 'Music',
'Educational', 'Casino', 'Adventure;Brain Games',
'Video Players & Editors;Music & Video',
'Entertainment;Creativity', 'Sports;Action & Adventure',
'Books & Reference;Education', 'Puzzle;Education',
'Role Playing;Brain Games', 'Strategy;Education',
'Racing;Pretend Play', 'Strategy;Creativity'], dtype=object)
```

There are too many categories under Genres. Hence, we will try to reduce some categories which have very few samples under them and put them under one new common category i.e. "Other"

Create an empty list

[63]:
```
lists = []
#Get the total genres count and gernes count of perticular gerner count less than 20 append those into the list
for i in inp2.Genres.value_counts().index:
    if inp2.Genres.value_counts()[i]<20:
        lists.append(i)
#changing the gerners which are in the list to other
```

```
inp2.Genres = ['Other' if i in lists else i for i in inp2.Genres]
inp2["Genres"].unique()
```

[63]:
```
array(['Art & Design', 'Other', 'Auto & Vehicles', 'Beauty',
       'Books & Reference', 'Business', 'Comics', 'Communication',
       'Dating', 'Education', 'Education;Education',
       'Education;Pretend Play', 'Entertainment', 'Events', 'Finance',
       'Food & Drink', 'Health & Fitness', 'House & Home',
       'Libraries & Demo', 'Lifestyle', 'Card', 'Casual', 'Puzzle',
       'Action', 'Arcade', 'Word', 'Racing', 'Sports', 'Board',
       'Simulation', 'Role Playing', 'Strategy', 'Trivia',
       'Educational;Education', 'Casual;Pretend Play', 'Medical',
       'Social', 'Shopping', 'Photography', 'Travel & Local', 'Tools',
       'Personalization', 'Productivity', 'Parenting', 'Weather',
       'Video Players & Editors', 'News & Magazines', 'Maps & Navigation',
       'Adventure', 'Educational', 'Casino'], dtype=object)
```

Storing the genres column into x varible and delete the genres col from dataframe inp2 And concat the encoded cols to the dataframe inp2

[64]:
```
inp2.Genres = pd.Categorical(inp2['Genres'])
x = inp2[["Genres"]]
del inp2['Genres']
dummies = pd.get_dummies(x, prefix = 'Genres')
inp2 = pd.concat([inp2,dummies], axis=1)
```

[65]:
```
inp2.head()
```

[65]:

	Rating	Reviews	Size	Installs	Price	Content Rating	\
0	4.1	5.075174	19.0	10000	0.0	Everyone	
1	3.9	6.875232	14.0	500000	0.0	Everyone	
2	4.7	11.379520	8.7	5000000	0.0	Everyone	
4	4.3	6.875232	2.8	100000	0.0	Everyone	
5	4.4	5.123964	5.6	50000	0.0	Everyone	

	Category_ART_AND_DESIGN	Category_AUTO_AND_VEHICLES	Category_BEAUTY	\
0	True	False	False	
1	True	False	False	
2	True	False	False	
4	True	False	False	
5	True	False	False	

	Category_BOOKS_AND_REFERENCE	…	Genres_Simulation	Genres_Social	\
0	False	…	False	False	
1	False	…	False	False	
2	False	…	False	False	
4	False	…	False	False	

```
5                        False  …              False          False

   Genres_Sports  Genres_Strategy  Genres_Tools  Genres_Travel & Local  \
0          False            False         False                  False
1          False            False         False                  False
2          False            False         False                  False
4          False            False         False                  False
5          False            False         False                  False

   Genres_Trivia  Genres_Video Players & Editors  Genres_Weather  Genres_Word
0          False                           False           False        False
1          False                           False           False        False
2          False                           False           False        False
4          False                           False           False        False
5          False                           False           False        False

[5 rows x 90 columns]
```

getting the unique values in Column "Content Rating"

[66]:
```
inp2["Content Rating"].unique()
```

[66]:
```
array(['Everyone', 'Teen', 'Everyone 10+', 'Mature 17+',
       'Adults only 18+', 'Unrated'], dtype=object)
```

Applying one hot encoding Storing the Content Rating column into x varible and delete the Content Rating col from dataframe inp2 And concat the encoded cols to the dataframe inp2

[67]:
```
inp2['Content Rating'] = pd.Categorical(inp2['Content Rating'])

x = inp2[['Content Rating']]
del inp2['Content Rating']

dummies = pd.get_dummies(x, prefix = 'Content Rating')
inp2 = pd.concat([inp2,dummies], axis=1)
inp2.head()
```

[67]:
```
   Rating    Reviews  Size  Installs  Price  Category_ART_AND_DESIGN  \
0     4.1   5.075174  19.0     10000    0.0                     True
1     3.9   6.875232  14.0    500000    0.0                     True
2     4.7  11.379520   8.7   5000000    0.0                     True
4     4.3   6.875232   2.8    100000    0.0                     True
5     4.4   5.123964   5.6     50000    0.0                     True

   Category_AUTO_AND_VEHICLES  Category_BEAUTY  Category_BOOKS_AND_REFERENCE  \
0                       False            False                         False
1                       False            False                         False
2                       False            False                         False
```

```
4                      False            False                          False
5                      False            False                          False

   Category_BUSINESS  …  Genres_Trivia  Genres_Video Players & Editors  \
0              False  …          False                           False
1              False  …          False                           False
2              False  …          False                           False
4              False  …          False                           False
5              False  …          False                           False

   Genres_Weather  Genres_Word  Content Rating_Adults only 18+  \
0           False        False                           False
1           False        False                           False
2           False        False                           False
4           False        False                           False
5           False        False                           False

   Content Rating_Everyone  Content Rating_Everyone 10+  \
0                     True                        False
1                     True                        False
2                     True                        False
4                     True                        False
5                     True                        False

   Content Rating_Mature 17+  Content Rating_Teen  Content Rating_Unrated
0                      False                False                   False
1                      False                False                   False
2                      False                False                   False
4                      False                False                   False
5                      False                False                   False

[5 rows x 95 columns]
```

[68]:
```
inp2.shape
```

[68]:
```
(7307, 95)
```

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

[69]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error as mse
from sklearn import metrics
```

[70]:
```
df2 = inp2
X = df2.drop('Rating',axis=1)
```

```
y = df2['Rating']

#Dividing the X and y into test and train data
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3,␣
 ↪random_state=5)
```

1.7 Model Building & Evaluation

Model building Use linear regression as the technique Report the R2 on the train set Create a linear reggression obj by calling the linear reggressor algorithm

[71]:
```
lin_reggressor = LinearRegression()
lin_reggressor.fit(X_train,y_train)
```

[71]:
```
LinearRegression()
```

[72]:
```
R2_Score_train_data = round(lin_reggressor.score(X_train,y_train),3)
print("The R2 value of the Training Set is : {}".format(R2_Score_train_data))
```

```
The R2 value of the Training Set is : 0.068
```

Make predictions on test set and report R2.

[73]:
```
y_pred = lin_reggressor.predict(X_test)
R2_Score_test_data =metrics.r2_score(y_test,y_pred)
R2_Score_test_data
```

[73]:
```
0.057753700325225865
```

[74]:
```
R2_Score_test_data = round(lin_reggressor.score(X_test,y_test),3)
print("The R2 value of the Training Set is : {}".format(R2_Score_test_data))
```

```
The R2 value of the Training Set is : 0.058
```

[ ]:
```

```