App Rating Prediction

Ankit Parashar

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.

1. Load the data file using pandas

In [1]:

```
# Importing the required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:

```
# Importing the data
df = pd.read_csv('googleplaystore.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone

In [4]:

```
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
```

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

In [5]:

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

Out[5]:

```
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
```

The column **Rating** has the most null values.

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

In [6]:

```
df.dropna(axis = 0, inplace=True)
```

In [7]:

```
df.reset_index(inplace=True)
```

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9360 entries, 0 to 9359
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   index           9360 non-null   int64  
 1   App             9360 non-null   object 
 2   Category        9360 non-null   object 
 3   Rating          9360 non-null   float64
 4   Reviews         9360 non-null   object 
 5   Size            9360 non-null   object 
 6   Installs        9360 non-null   object 
 7   Type            9360 non-null   object 
 8   Price           9360 non-null   object 
 9   Content Rating  9360 non-null   object 
 10  Genres          9360 non-null   object 
 11  Last Updated    9360 non-null   object 
 12  Current Ver     9360 non-null   object 
 13  Android Ver     9360 non-null   object 
dtypes: float64(1), int64(1), object(12)
memory usage: 1023.9+ KB
```

The data frame is now free from any null records.

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.

1. Extract the numeric value from the column

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

Let us first check if the data frame has any value apart from the ones mentioned in MB or kB. We will then check how to treat those.

In [9]:

```
df[(df['Size'].str.endswith('M') == False) & (df['Size'].str.endswith('k') == False)][
'Size'].value_counts()
```

Out[9]:

```
Varies with device    1637
Name: Size, dtype: int64
```

There is a value **Varies with device** on 1637 records. We will need to remove these records as these are as good as null values for us.

In [10]:

```
df = df[df['Size'] != 'Varies with device']
```

In [11]:

```
df[(df['Size'].str.endswith('M') == False) & (df['Size'].str.endswith('k') == False)][
'Size'].value_counts()
```

Out[11]:

```
Series([], Name: Size, dtype: int64)
```

We will now tackle the problem at hand.

In [12]:

```
df['Size'] = df['Size'].apply(lambda x: float(x.split('M')[0]) * 1000 if x.endswith('M'
) else float(x.split('k')[0]))
```

In [13]:

```
df['Size'].head()
```

Out[13]:

```
0    19000.0
1    14000.0
2     8700.0
3    25000.0
4     2800.0
Name: Size, dtype: float64
```

The **Size** column is therefore transformed into a numeric type with values corrected.

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

In [14]:

```
df['Reviews'] = df['Reviews'].astype('int64')
```

In [15]:

```
df['Reviews'].head()
```

Out[15]:

```
0       159
1       967
2     87510
3    215644
4       967
Name: Reviews, dtype: int64
```

Reviews column is converted to 64 bit integer

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

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

1. Remove '+', ',' from the field, convert it to integer

In [16]:

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

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

In [17]:

```
df['Price'] = df['Price'].apply(lambda x: float(x.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 [18]:

```
df[(df['Rating'] < 1) | (df['Rating'] > 5)]
```

Out[18]:

index	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Upd

No such record exists where the **Rating** is **<1 or >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 [19]:

```
df[df['Reviews'] > df['Installs']]
```

Out[19]:

	index	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	C
2340	2454	KBA-EZ Health Guide	MEDICAL	5.0	4	25000.0	1	Free	0.00	Everyone	M
5535	5917	Ra Ga Ba	GAME	5.0	2	20000.0	1	Paid	1.49	Everyone	
6144	6700	Brick Breaker BR	GAME	5.0	7	19000.0	5	Free	0.00	Everyone	
6616	7402	Trovami se ci riesci	GAME	5.0	11	6100.0	10	Free	0.00	Everyone	
7592	8591	DN Blog	SOCIAL	5.0	20	4200.0	10	Free	0.00	Teen	
9260	10697	Mu.F.O.	GAME	5.0	2	16000.0	1	Paid	0.99	Everyone	

In [20]:

```
df.drop(df.loc[df['Reviews'] > df['Installs']].index, axis = 0, inplace=True)
```

We will also drop the unneccessary column index as we already reset it.

In [21]:

```
df.drop(['index'], axis = 1, inplace=True)
```

In [22]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7717 entries, 0 to 9359
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             7717 non-null   object 
 1   Category        7717 non-null   object 
 2   Rating          7717 non-null   float64
 3   Reviews         7717 non-null   int64  
 4   Size            7717 non-null   float64
 5   Installs        7717 non-null   int64  
 6   Type            7717 non-null   object 
 7   Price           7717 non-null   float64
 8   Content Rating  7717 non-null   object 
 9   Genres          7717 non-null   object 
 10  Last Updated    7717 non-null   object 
 11  Current Ver     7717 non-null   object 
 12  Android Ver     7717 non-null   object 
dtypes: float64(3), int64(2), object(8)
memory usage: 844.0+ KB
```

In [23]:

```
df[(df['Type'] == 'Free') & (df['Price'] >0)]
```

Out[23]:

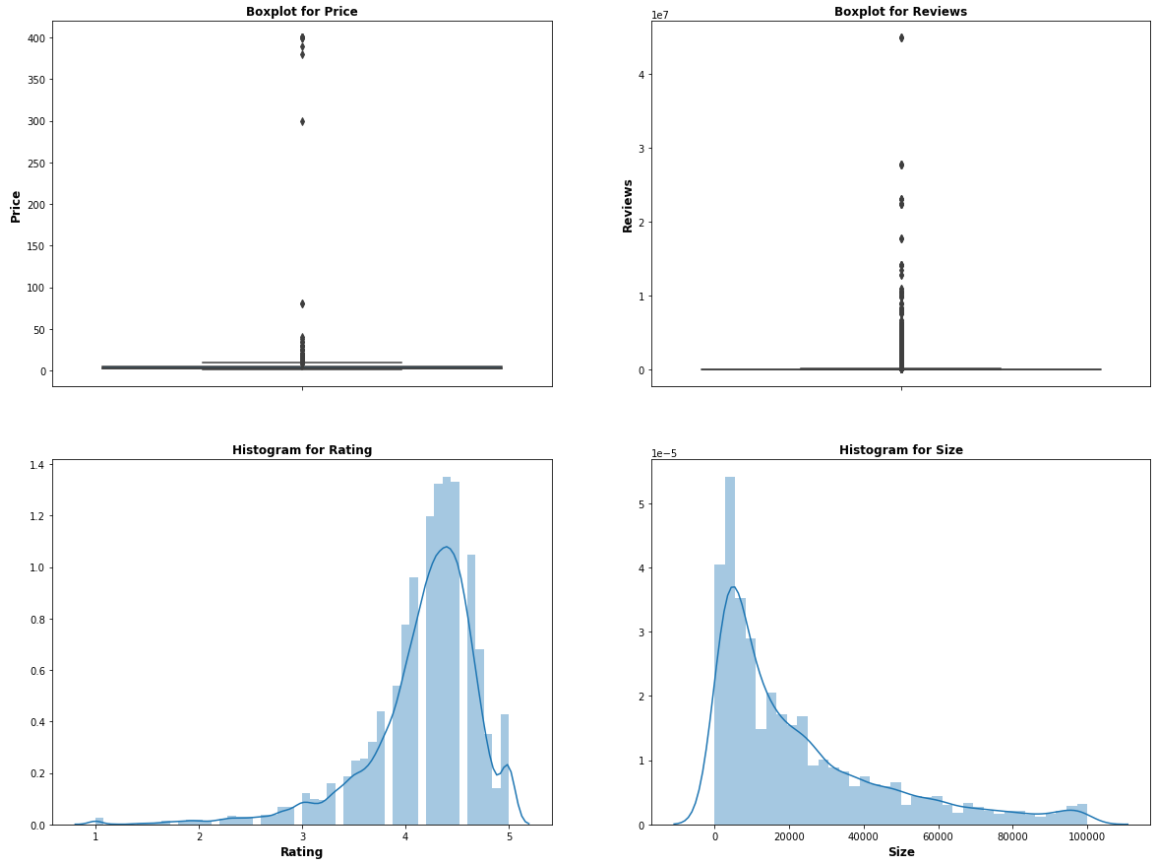
App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	C

No such record exists where the app **Type** is **Free** and **Price** is **>0**

6. Performing univariate analysis:

In [24]:

```
fig, axes = plt.subplots(2, 2, figsize = (20, 15))
axes[0, 0].set_title('Boxplot for Price', fontsize = 12, fontweight = 'semibold')
axes[0, 0].set_ylabel('Price', fontsize = 12, fontweight = 'semibold')
sns.boxplot(y = 'Price', data = df[df['Price'] != 0], ax = axes[0, 0])
axes[0, 1].set_title('Boxplot for Reviews', fontsize = 12, fontweight = 'semibold')
axes[0, 1].set_ylabel('Reviews', fontsize = 12, fontweight = 'semibold')
sns.boxplot(y = 'Reviews', data = df, ax = axes[0, 1])
axes[1, 0].set_title('Histogram for Rating', fontsize = 12, fontweight = 'semibold')
axes[1, 0].set_xlabel('Ratings', fontsize = 12, fontweight = 'semibold')
sns.distplot(df['Rating'], ax = axes[1, 0])
axes[1, 1].set_title('Histogram for Size', fontsize = 12, fontweight = 'semibold')
axes[1, 1].set_xlabel('Size', fontsize = 12, fontweight = 'semibold')
sns.distplot(df['Size'], ax = axes[1, 1]);
```



We can infer the following by observing the

* Price of most of the apps seems to be nominal. It is mostly a little over 0 USD for a large number of apps.

* There are a few apps that have an unusually large number of reviews.

* The rating for maximum number of apps is concentrated between 4 & 5

* The most common size of the apps is around 5-10 MB

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!

We will drop all records where the price is >= 200$

In [25]:

```
df.loc[df['Price'] >=200]
```

Out[25]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
4036	most expensive app (H)	FAMILY	4.3	6	1500.0	100	Paid	399.99	Everyone
4189	💎 I'm rich	LIFESTYLE	3.8	718	26000.0	10000	Paid	399.99	Everyone
4194	I'm Rich - Trump Edition	LIFESTYLE	3.6	275	7300.0	10000	Paid	400.00	Everyone
5042	I am rich	LIFESTYLE	3.8	3547	1800.0	100000	Paid	399.99	Everyone
5045	I am Rich Plus	FAMILY	4.0	856	8700.0	10000	Paid	399.99	Everyone
5046	I am rich VIP	LIFESTYLE	3.8	411	2600.0	10000	Paid	299.99	Everyone
5047	I Am Rich Premium	FINANCE	4.1	1867	4700.0	50000	Paid	399.99	Everyone
5048	I am extremely Rich	LIFESTYLE	2.9	41	2900.0	1000	Paid	379.99	Everyone
5049	I am Rich!	FINANCE	3.8	93	22000.0	1000	Paid	399.99	Everyone
5050	I am rich(premium)	FINANCE	3.5	472	965.0	5000	Paid	399.99	Everyone
5053	I Am Rich Pro	FAMILY	4.4	201	2700.0	5000	Paid	399.99	Everyone
5055	I am rich (Most expensive app)	FINANCE	4.1	129	2700.0	1000	Paid	399.99	Teen
5057	I Am Rich	FAMILY	3.6	217	4900.0	10000	Paid	389.99	Everyone
5060	I am Rich	FINANCE	4.3	180	3800.0	5000	Paid	399.99	Everyone
5064	I AM RICH PRO PLUS	FINANCE	4.0	36	41000.0	1000	Paid	399.99	Everyone

In [26]:

```
df.drop(df.loc[df['Price'] >=200].index, axis = 0, inplace=True)
```

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 [27]:

```
df.loc[df['Reviews'] >=2000000]
```

Out[27]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Cor Ra
332	Yahoo Mail – Stay Organized	COMMUNICATION	4.3	4187998	16000.0	100000000	Free	0.0	Ever
334	imo free video calls and chat	COMMUNICATION	4.3	4785892	11000.0	500000000	Free	0.0	Ever
353	UC Browser Mini -Tiny Fast Private & Secure	COMMUNICATION	4.4	3648120	3300.0	100000000	Free	0.0	
365	UC Browser - Fast Download Private & Secure	COMMUNICATION	4.5	17712922	40000.0	500000000	Free	0.0	
370	imo free video calls and chat	COMMUNICATION	4.3	4785988	11000.0	500000000	Free	0.0	Ever
...	...	...	...	...	...	...	...	...	
8053	Need for Speed™ No Limits	GAME	4.4	3344300	22000.0	50000000	Free	0.0	Ever
8076	Modern Combat 5: eSports FPS	GAME	4.3	2903386	58000.0	100000000	Free	0.0	Ma
8883	Farm Heroes Saga	FAMILY	4.4	7615646	71000.0	100000000	Free	0.0	Ever
8886	Fallout Shelter	FAMILY	4.6	2721923	25000.0	10000000	Free	0.0	
9015	Garena Free Fire	GAME	4.5	5534114	53000.0	100000000	Free	0.0	

219 rows × 13 columns

In [28]:

```
df.drop(df.loc[df['Reviews'] >=2000000].index, axis = 0, inplace=True)
```

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.

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

In [29]:

```
df[['Installs']].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

Out[29]:

	Installs
0.10	1000.0
0.25	10000.0
0.50	100000.0
0.70	1000000.0
0.90	10000000.0
0.95	10000000.0
0.99	50000000.0

Table above shows the distribution in terms of the mentioned quantiles for the column **Installs**

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

In [30]:

```
len(df[df['Installs'] > 50000000.0])
```

Out[30]:

```
60
```

There seem to be only very few rows with **Installs** > 50000000. We will get rid of these rows as they are most probably outliers in our data

In [31]:

```
df = df[df['Installs'] < 50000000.0]
```

In [32]:

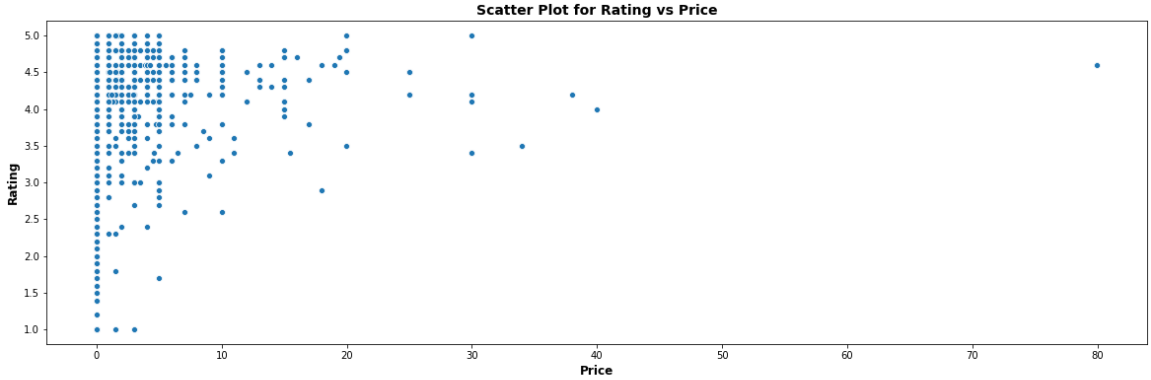
```
df.info()
```

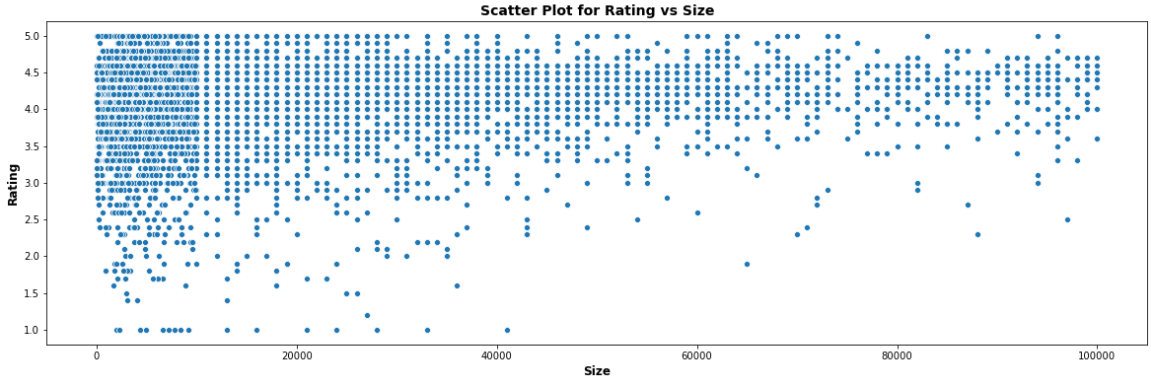
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7307 entries, 0 to 9359
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             7307 non-null   object 
 1   Category        7307 non-null   object 
 2   Rating          7307 non-null   float64
 3   Reviews         7307 non-null   int64  
 4   Size            7307 non-null   float64
 5   Installs        7307 non-null   int64  
 6   Type            7307 non-null   object 
 7   Price           7307 non-null   float64
 8   Content Rating  7307 non-null   object 
 9   Genres          7307 non-null   object 
 10  Last Updated    7307 non-null   object 
 11  Current Ver     7307 non-null   object 
 12  Android Ver     7307 non-null   object 
dtypes: float64(3), int64(2), object(8)
memory usage: 799.2+ KB
```

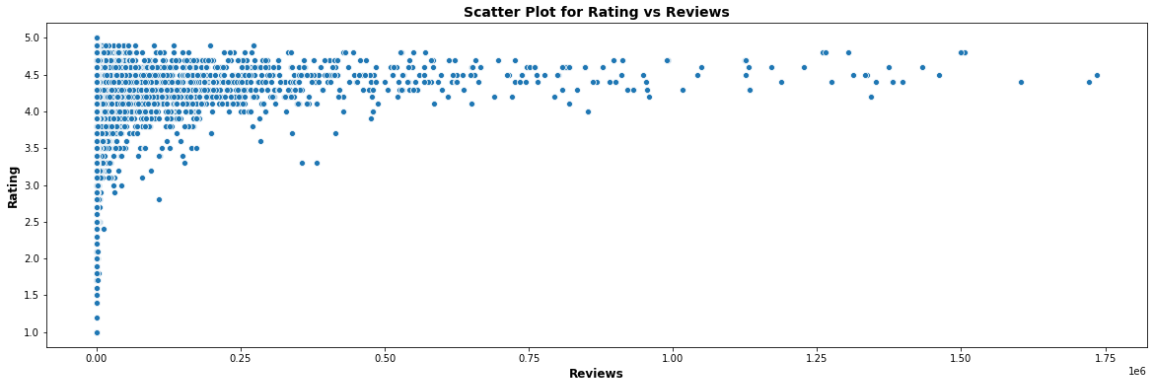
8. Bivariate analysis:

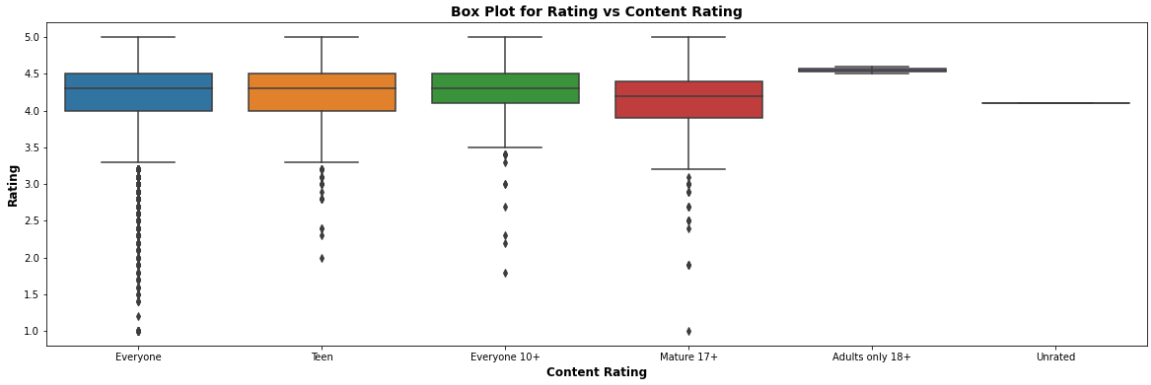
In [33]:

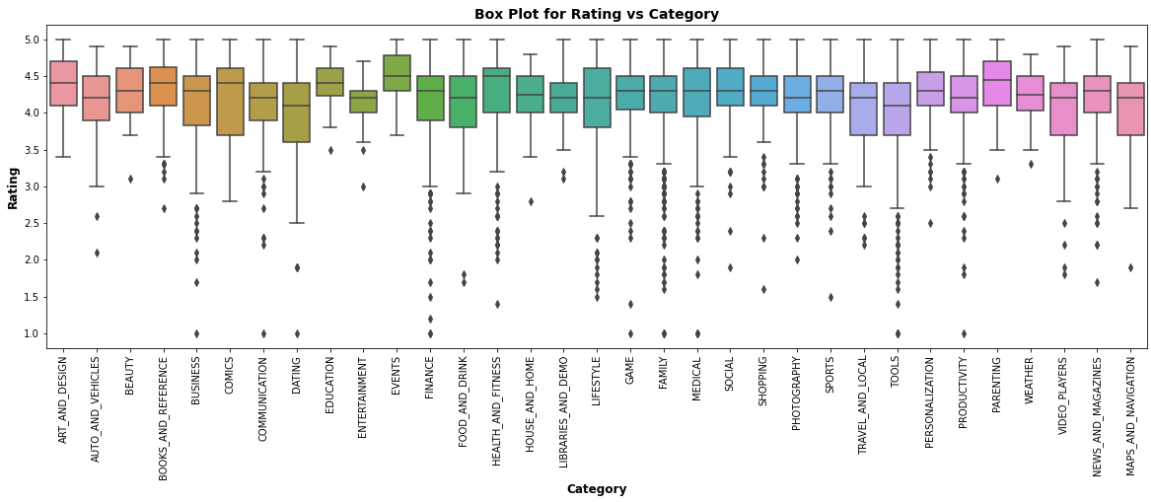
```
fig, axes = plt.subplots(5, 1, figsize = (20, 35))
axes[0].set_title('Scatter Plot for Rating vs Price', fontsize = 14, fontweight = 'semi
bold')
axes[0].set_xlabel('Price', fontsize = 12, fontweight = 'semibold')
axes[0].set_ylabel('Rating', fontsize = 12, fontweight = 'semibold')
sns.scatterplot(x = 'Price', y = 'Rating', data = df, ax = axes[0])
axes[1].set_title('Scatter Plot for Rating vs Size', fontsize = 14, fontweight = 'semib
old')
axes[1].set_xlabel('Size', fontsize = 12, fontweight = 'semibold')
axes[1].set_ylabel('Rating', fontsize = 12, fontweight = 'semibold')
sns.scatterplot(x = 'Size', y = 'Rating', data = df, ax = axes[1])
axes[2].set_title('Scatter Plot for Rating vs Reviews', fontsize = 14, fontweight = 'se
mibold')
axes[2].set_xlabel('Reviews', fontsize = 12, fontweight = 'semibold')
axes[2].set_ylabel('Rating', fontsize = 12, fontweight = 'semibold')
sns.scatterplot(x = 'Reviews', y = 'Rating', data = df, ax = axes[2])
axes[3].set_title('Box Plot for Rating vs Content Rating', fontsize = 14, fontweight =
'semibold')
axes[3].set_xlabel('Content Rating', fontsize = 12, fontweight = 'semibold')
axes[3].set_ylabel('Rating', fontsize = 12, fontweight = 'semibold')
sns.boxplot(x = 'Content Rating', y = 'Rating', data = df, ax = axes[3])
axes[4].set_title('Box Plot for Rating vs Category', fontsize = 14, fontweight = 'semib
old')
axes[4].set_xlabel('Category', fontsize = 12, fontweight = 'semibold')
axes[4].set_ylabel('Rating', fontsize = 12, fontweight = 'semibold')
plt.xticks(rotation = 90)
sns.boxplot(x = 'Category', y = 'Rating', data = df, ax = axes[4]);
```


Scatter Plot for Rating vs Price
Rating
Price


Scatter Plot for Rating vs Size
Rating
Size


Scatter Plot for Rating vs Reviews
Rating
Reviews
1e6


Box Plot for Rating vs Content Rating
Rating
Everyone
Teen
Everyone 10+
Mature 17+
Adults only 18+
Unrated
Content Rating


Box Plot for Rating vs Category
Rating
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
Category

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

No. Rating does not increase with price. On the contrary, the free and relatively cheap apps are seen to have better ratings and are also installed more than pricier apps.

2. Are heavier apps rated better?

There are a few apps which are heavy and are highly rated. But in general there is no such trend that heavier apps are rated better.

3. Does more review mean a better rating always?

Not always. There are apps that haven't been reviewed by are rated quite well.

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

From the box plot, we can infer that apps with Content Rating **Everyone 10+** & **Adults only 18+** have better rating generally.

5. Which genre has the best ratings?

Categories which have good ratings:

1. BEAUTY
2. COMICS
3. EDUCATION
4. EVENTS
5. LIBRARIES_AND_DEMO
6. WEATHER

9. Data Preprocessing

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

In [34]:

```
inp1 = df
```

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 [35]:

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

In [36]:

```
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 [37]:

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

In [38]:

```
inp1.head()
```

Out[38]:

	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
0	ART_AND_DESIGN	4.1	5.075174	19000.0	9.210440	Free	0.0	Everyone	Art &
1	ART_AND_DESIGN	3.9	6.875232	14000.0	13.122365	Free	0.0	Everyone	Design;
2	ART_AND_DESIGN	4.7	11.379520	8700.0	15.424949	Free	0.0	Everyone	Art &
4	ART_AND_DESIGN	4.3	6.875232	2800.0	11.512935	Free	0.0	Everyone	Design;C
5	ART_AND_DESIGN	4.4	5.123964	5600.0	10.819798	Free	0.0	Everyone	Art &

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 [39]:

```
category = pd.get_dummies(inp1['Category'])
```

In [40]:

```
category.head()
```

Out[40]:

	ART_AND_DESIGN	AUTO_AND_VEHICLES	BEAUTY	BOOKS_AND_REFERENCE	BUSINES
0	1	0	0	0	
1	1	0	0	0	
2	1	0	0	0	
4	1	0	0	0	
5	1	0	0	0	

5 rows × 33 columns

In [41]:

```
genres = inp1['Genres'].str.get_dummies(';')
```

In [42]:

```
genres
```

Out[42]:

	Action	Action & Adventure	Adventure	Arcade	Art & Design	Auto & Vehicles	Beauty	Board	Books & Reference	C
0	0	0	0	0	1	0	0	0	0	
1	0	0	0	0	1	0	0	0	0	
2	0	0	0	0	1	0	0	0	0	
4	0	0	0	0	1	0	0	0	0	
5	0	0	0	0	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
9354	0	0	0	0	0	0	0	0	1	
9355	0	0	0	0	0	0	0	0	0	
9356	0	0	0	0	0	0	0	0	0	
9357	0	0	0	0	0	0	0	0	0	
9359	0	0	0	0	0	0	0	0	0	

7307 rows × 53 columns

In [43]:

```
content = pd.get_dummies(inp1['Content Rating'])
```

In [44]:

```
content.head()
```

Out[44]:

	Adults only 18+	Everyone	Everyone 10+	Mature 17+	Teen	Unrated
0	0	1	0	0	0	0
1	0	1	0	0	0	0
2	0	1	0	0	0	0
4	0	1	0	0	0	0
5	0	1	0	0	0	0

In [45]:

```
cat_features = pd.concat([category, genres, content], axis = 1)
```

In [46]:

```
cat_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7307 entries, 0 to 9359
Data columns (total 92 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ART_AND_DESIGN           7307 non-null   uint8
 1   AUTO_AND_VEHICLES        7307 non-null   uint8
 2   BEAUTY                   7307 non-null   uint8
 3   BOOKS_AND_REFERENCE      7307 non-null   uint8
 4   BUSINESS                 7307 non-null   uint8
 5   COMICS                   7307 non-null   uint8
 6   COMMUNICATION            7307 non-null   uint8
 7   DATING                   7307 non-null   uint8
 8   EDUCATION                7307 non-null   uint8
 9   ENTERTAINMENT            7307 non-null   uint8
 10  EVENTS                   7307 non-null   uint8
 11  FAMILY                   7307 non-null   uint8
 12  FINANCE                  7307 non-null   uint8
 13  FOOD_AND_DRINK           7307 non-null   uint8
 14  GAME                     7307 non-null   uint8
 15  HEALTH_AND_FITNESS       7307 non-null   uint8
 16  HOUSE_AND_HOME           7307 non-null   uint8
 17  LIBRARIES_AND_DEMO       7307 non-null   uint8
 18  LIFESTYLE                7307 non-null   uint8
 19  MAPS_AND_NAVIGATION      7307 non-null   uint8
 20  MEDICAL                  7307 non-null   uint8
 21  NEWS_AND_MAGAZINES       7307 non-null   uint8
 22  PARENTING                7307 non-null   uint8
 23  PERSONALIZATION          7307 non-null   uint8
 24  PHOTOGRAPHY              7307 non-null   uint8
 25  PRODUCTIVITY             7307 non-null   uint8
 26  SHOPPING                 7307 non-null   uint8
 27  SOCIAL                   7307 non-null   uint8
 28  SPORTS                   7307 non-null   uint8
 29  TOOLS                    7307 non-null   uint8
 30  TRAVEL_AND_LOCAL         7307 non-null   uint8
 31  VIDEO_PLAYERS            7307 non-null   uint8
 32  WEATHER                  7307 non-null   uint8
 33  Action                   7307 non-null   int64
 34  Action & Adventure       7307 non-null   int64
 35  Adventure                7307 non-null   int64
 36  Arcade                   7307 non-null   int64
 37  Art & Design             7307 non-null   int64
 38  Auto & Vehicles          7307 non-null   int64
 39  Beauty                   7307 non-null   int64
 40  Board                    7307 non-null   int64
 41  Books & Reference        7307 non-null   int64
 42  Brain Games              7307 non-null   int64
 43  Business                 7307 non-null   int64
 44  Card                     7307 non-null   int64
 45  Casino                   7307 non-null   int64
 46  Casual                   7307 non-null   int64
 47  Comics                   7307 non-null   int64
 48  Communication            7307 non-null   int64
 49  Creativity               7307 non-null   int64
 50  Dating                   7307 non-null   int64
 51  Education                7307 non-null   int64
 52  Educational              7307 non-null   int64
 53  Entertainment            7307 non-null   int64
 54  Events                   7307 non-null   int64
 55  Finance                  7307 non-null   int64
```

```
 56  Food & Drink             7307 non-null   int64
 57  Health & Fitness         7307 non-null   int64
 58  House & Home             7307 non-null   int64
 59  Libraries & Demo         7307 non-null   int64
 60  Lifestyle                7307 non-null   int64
 61  Maps & Navigation        7307 non-null   int64
 62  Medical                  7307 non-null   int64
 63  Music                    7307 non-null   int64
 64  Music & Audio            7307 non-null   int64
 65  Music & Video            7307 non-null   int64
 66  News & Magazines         7307 non-null   int64
 67  Parenting                7307 non-null   int64
 68  Personalization          7307 non-null   int64
 69  Photography              7307 non-null   int64
 70  Pretend Play             7307 non-null   int64
 71  Productivity             7307 non-null   int64
 72  Puzzle                   7307 non-null   int64
 73  Racing                   7307 non-null   int64
 74  Role Playing             7307 non-null   int64
 75  Shopping                 7307 non-null   int64
 76  Simulation               7307 non-null   int64
 77  Social                   7307 non-null   int64
 78  Sports                   7307 non-null   int64
 79  Strategy                 7307 non-null   int64
 80  Tools                    7307 non-null   int64
 81  Travel & Local           7307 non-null   int64
 82  Trivia                   7307 non-null   int64
 83  Video Players & Editors  7307 non-null   int64
 84  Weather                  7307 non-null   int64
 85  Word                     7307 non-null   int64
 86  Adults only 18+          7307 non-null   uint8
 87  Everyone                 7307 non-null   uint8
 88  Everyone 10+             7307 non-null   uint8
 89  Mature 17+               7307 non-null   uint8
 90  Teen                     7307 non-null   uint8
 91  Unrated                  7307 non-null   uint8
dtypes: int64(53), uint8(39)
memory usage: 3.6 MB
```

Regularizing the dataset for all the categorical features.

In [47]:

```
cat_features = cat_features.astype('int64')
```

In [48]:

```
cat_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7307 entries, 0 to 9359
Data columns (total 92 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ART_AND_DESIGN           7307 non-null   int64
 1   AUTO_AND_VEHICLES        7307 non-null   int64
 2   BEAUTY                   7307 non-null   int64
 3   BOOKS_AND_REFERENCE      7307 non-null   int64
 4   BUSINESS                 7307 non-null   int64
 5   COMICS                   7307 non-null   int64
 6   COMMUNICATION            7307 non-null   int64
 7   DATING                   7307 non-null   int64
 8   EDUCATION                7307 non-null   int64
 9   ENTERTAINMENT            7307 non-null   int64
 10  EVENTS                   7307 non-null   int64
 11  FAMILY                   7307 non-null   int64
 12  FINANCE                  7307 non-null   int64
 13  FOOD_AND_DRINK           7307 non-null   int64
 14  GAME                     7307 non-null   int64
 15  HEALTH_AND_FITNESS       7307 non-null   int64
 16  HOUSE_AND_HOME           7307 non-null   int64
 17  LIBRARIES_AND_DEMO       7307 non-null   int64
 18  LIFESTYLE                7307 non-null   int64
 19  MAPS_AND_NAVIGATION      7307 non-null   int64
 20  MEDICAL                  7307 non-null   int64
 21  NEWS_AND_MAGAZINES       7307 non-null   int64
 22  PARENTING                7307 non-null   int64
 23  PERSONALIZATION          7307 non-null   int64
 24  PHOTOGRAPHY              7307 non-null   int64
 25  PRODUCTIVITY             7307 non-null   int64
 26  SHOPPING                 7307 non-null   int64
 27  SOCIAL                   7307 non-null   int64
 28  SPORTS                   7307 non-null   int64
 29  TOOLS                    7307 non-null   int64
 30  TRAVEL_AND_LOCAL         7307 non-null   int64
 31  VIDEO_PLAYERS            7307 non-null   int64
 32  WEATHER                  7307 non-null   int64
 33  Action                   7307 non-null   int64
 34  Action & Adventure       7307 non-null   int64
 35  Adventure                7307 non-null   int64
 36  Arcade                   7307 non-null   int64
 37  Art & Design             7307 non-null   int64
 38  Auto & Vehicles          7307 non-null   int64
 39  Beauty                   7307 non-null   int64
 40  Board                    7307 non-null   int64
 41  Books & Reference        7307 non-null   int64
 42  Brain Games              7307 non-null   int64
 43  Business                 7307 non-null   int64
 44  Card                     7307 non-null   int64
 45  Casino                   7307 non-null   int64
 46  Casual                   7307 non-null   int64
 47  Comics                   7307 non-null   int64
 48  Communication            7307 non-null   int64
 49  Creativity               7307 non-null   int64
 50  Dating                   7307 non-null   int64
 51  Education                7307 non-null   int64
 52  Educational              7307 non-null   int64
 53  Entertainment            7307 non-null   int64
 54  Events                   7307 non-null   int64
 55  Finance                  7307 non-null   int64
```

```
 56  Food & Drink             7307 non-null   int64
 57  Health & Fitness         7307 non-null   int64
 58  House & Home             7307 non-null   int64
 59  Libraries & Demo         7307 non-null   int64
 60  Lifestyle                7307 non-null   int64
 61  Maps & Navigation        7307 non-null   int64
 62  Medical                  7307 non-null   int64
 63  Music                    7307 non-null   int64
 64  Music & Audio            7307 non-null   int64
 65  Music & Video            7307 non-null   int64
 66  News & Magazines         7307 non-null   int64
 67  Parenting                7307 non-null   int64
 68  Personalization          7307 non-null   int64
 69  Photography              7307 non-null   int64
 70  Pretend Play             7307 non-null   int64
 71  Productivity             7307 non-null   int64
 72  Puzzle                   7307 non-null   int64
 73  Racing                   7307 non-null   int64
 74  Role Playing             7307 non-null   int64
 75  Shopping                 7307 non-null   int64
 76  Simulation               7307 non-null   int64
 77  Social                   7307 non-null   int64
 78  Sports                   7307 non-null   int64
 79  Strategy                 7307 non-null   int64
 80  Tools                    7307 non-null   int64
 81  Travel & Local           7307 non-null   int64
 82  Trivia                   7307 non-null   int64
 83  Video Players & Editors  7307 non-null   int64
 84  Weather                  7307 non-null   int64
 85  Word                     7307 non-null   int64
 86  Adults only 18+          7307 non-null   int64
 87  Everyone                 7307 non-null   int64
 88  Everyone 10+             7307 non-null   int64
 89  Mature 17+               7307 non-null   int64
 90  Teen                     7307 non-null   int64
 91  Unrated                  7307 non-null   int64
dtypes: int64(92)
memory usage: 5.5 MB
```

In [49]:

```
inp2 = pd.concat([inp1, cat_features], axis = 1).drop(['Category', 'Genres', 'Content Rating', 'Type'], axis = 1).reset_index()
```

In [50]:

```
inp2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7307 entries, 0 to 7306
Data columns (total 98 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   index                   7307 non-null   int64  
 1   Rating                  7307 non-null   float64
 2   Reviews                 7307 non-null   float64
 3   Size                    7307 non-null   float64
 4   Installs                7307 non-null   float64
 5   Price                   7307 non-null   float64
 6   ART_AND_DESIGN          7307 non-null   int64  
 7   AUTO_AND_VEHICLES       7307 non-null   int64  
 8   BEAUTY                  7307 non-null   int64  
 9   BOOKS_AND_REFERENCE     7307 non-null   int64  
 10  BUSINESS                7307 non-null   int64  
 11  COMICS                  7307 non-null   int64  
 12  COMMUNICATION           7307 non-null   int64  
 13  DATING                  7307 non-null   int64  
 14  EDUCATION               7307 non-null   int64  
 15  ENTERTAINMENT           7307 non-null   int64  
 16  EVENTS                  7307 non-null   int64  
 17  FAMILY                  7307 non-null   int64  
 18  FINANCE                 7307 non-null   int64  
 19  FOOD_AND_DRINK          7307 non-null   int64  
 20  GAME                    7307 non-null   int64  
 21  HEALTH_AND_FITNESS      7307 non-null   int64  
 22  HOUSE_AND_HOME          7307 non-null   int64  
 23  LIBRARIES_AND_DEMO      7307 non-null   int64  
 24  LIFESTYLE               7307 non-null   int64  
 25  MAPS_AND_NAVIGATION     7307 non-null   int64  
 26  MEDICAL                 7307 non-null   int64  
 27  NEWS_AND_MAGAZINES      7307 non-null   int64  
 28  PARENTING               7307 non-null   int64  
 29  PERSONALIZATION         7307 non-null   int64  
 30  PHOTOGRAPHY             7307 non-null   int64  
 31  PRODUCTIVITY            7307 non-null   int64  
 32  SHOPPING                7307 non-null   int64  
 33  SOCIAL                  7307 non-null   int64  
 34  SPORTS                  7307 non-null   int64  
 35  TOOLS                   7307 non-null   int64  
 36  TRAVEL_AND_LOCAL        7307 non-null   int64  
 37  VIDEO_PLAYERS           7307 non-null   int64  
 38  WEATHER                 7307 non-null   int64  
 39  Action                  7307 non-null   int64  
 40  Action & Adventure      7307 non-null   int64  
 41  Adventure               7307 non-null   int64  
 42  Arcade                  7307 non-null   int64  
 43  Art & Design            7307 non-null   int64  
 44  Auto & Vehicles         7307 non-null   int64  
 45  Beauty                  7307 non-null   int64  
 46  Board                   7307 non-null   int64  
 47  Books & Reference       7307 non-null   int64  
 48  Brain Games             7307 non-null   int64  
 49  Business                7307 non-null   int64  
 50  Card                    7307 non-null   int64  
 51  Casino                  7307 non-null   int64  
 52  Casual                  7307 non-null   int64  
 53  Comics                  7307 non-null   int64  
 54  Communication           7307 non-null   int64  
 55  Creativity              7307 non-null   int64  
```

```
 56  Dating                    7307 non-null   int64
 57  Education                 7307 non-null   int64
 58  Educational               7307 non-null   int64
 59  Entertainment             7307 non-null   int64
 60  Events                    7307 non-null   int64
 61  Finance                   7307 non-null   int64
 62  Food & Drink              7307 non-null   int64
 63  Health & Fitness          7307 non-null   int64
 64  House & Home              7307 non-null   int64
 65  Libraries & Demo          7307 non-null   int64
 66  Lifestyle                 7307 non-null   int64
 67  Maps & Navigation         7307 non-null   int64
 68  Medical                   7307 non-null   int64
 69  Music                     7307 non-null   int64
 70  Music & Audio             7307 non-null   int64
 71  Music & Video             7307 non-null   int64
 72  News & Magazines          7307 non-null   int64
 73  Parenting                 7307 non-null   int64
 74  Personalization           7307 non-null   int64
 75  Photography               7307 non-null   int64
 76  Pretend Play              7307 non-null   int64
 77  Productivity              7307 non-null   int64
 78  Puzzle                    7307 non-null   int64
 79  Racing                    7307 non-null   int64
 80  Role Playing              7307 non-null   int64
 81  Shopping                  7307 non-null   int64
 82  Simulation                7307 non-null   int64
 83  Social                    7307 non-null   int64
 84  Sports                    7307 non-null   int64
 85  Strategy                  7307 non-null   int64
 86  Tools                     7307 non-null   int64
 87  Travel & Local            7307 non-null   int64
 88  Trivia                    7307 non-null   int64
 89  Video Players & Editors   7307 non-null   int64
 90  Weather                   7307 non-null   int64
 91  Word                      7307 non-null   int64
 92  Adults only 18+           7307 non-null   int64
 93  Everyone                  7307 non-null   int64
 94  Everyone 10+              7307 non-null   int64
 95  Mature 17+                7307 non-null   int64
 96  Teen                      7307 non-null   int64
 97  Unrated                   7307 non-null   int64
dtypes: float64(5), int64(93)
memory usage: 5.5 MB
```

In [51]:

```
inp2.drop(['index'], 1, inplace=True)
```

10. 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.

In [52]:

```
X_train, X_test, y_train, y_test = train_test_split(inp2.drop(['Rating'], 1), inp2['Rat
ing'], test_size = 0.3, random_state = 42)
```

In [53]:

```
print('Shape of training features:', X_train.shape)
print('Shape of training output:', y_train.shape)
print('Shape of testing features', X_test.shape)
print('Shape of testing output', y_test.shape)
```

```
Shape of training features: (5114, 96)
Shape of training output: (5114,)
Shape of testing features (2193, 96)
Shape of testing output (2193,)
```

11. Model Building

* *Use linear regression as the technique*

In [54]:

```
lin_reg = LinearRegression()
```

In [55]:

```
lin_reg.fit(X_train, y_train)
```

Out[55]:

```
LinearRegression()
```

* *Report the R2 on the train set*

In [56]:

```
print('R2 on the training set is:', round(lin_reg.score(X_train, y_train), 4))
```

```
R2 on the training set is: 0.1537
```

12. Make predictions on test set and report R2.

In [57]:

```
y_pred = lin_reg.predict(X_test)
```

In [58]:

```
print('R2 on the test set is:', round(lin_reg.score(X_test, y_test), 4))
```

```
R2 on the test set is: 0.1222
```