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:

1. Load the data file using pandas.
2. Check for null values in the data. Get the number of null values for each column.
3. Drop records with nulls in any of the columns.
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
    2. Multiply the value by 1,000, if size is mentioned in Mb
2. Reviews is a numeric field that is loaded as a string field. Convert it to numeric (int/float).
3. Installs field is currently stored as string and has values like 1,000,000+.
    1. Treat 1,000,000+ as 1,000,000
    2. remove '+', ',' from the field, convert it to integer
4. Price field is a string and has $ symbol. Remove '$' sign, and convert it to numeric.

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.
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.
3. 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:

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!
    1. Check out the records with very high price
        1. Is 200 indeed a high price?
    2. Drop these as most seem to be junk apps
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.

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
    2. 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.

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

Steps to perform:

1. Load the data file using pandas.
2. Check for null values in the data. Get the number of null values for each column.
3. Drop records with nulls in any of the columns.
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
    2. Multiply the value by 1,000, if size is mentioned in Mb
2. Reviews is a numeric field that is loaded as a string field. Convert it to numeric (int/float).
3. Installs field is currently stored as string and has values like 1,000,000+.
    1. Treat 1,000,000+ as 1,000,000
    2. remove '+', ',' from the field, convert it to integer
4. Price field is a string and has $ symbol. Remove '$' sign, and convert it to numeric.

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.
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.
3. 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:

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!
    1. Check out the records with very high price
        1. Is 200 indeed a high price?
    2. Drop these as most seem to be junk apps
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.
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
    2. 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.

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

1.
    1. Are heavier apps rated better?
2. Make scatter plot/joinplot for Rating vs. Reviews
    1. Does more review mean a better rating always?
3. Make boxplot for Rating vs. Content Rating
    1. Is there any difference in the ratings? Are some types liked better?
4. Make boxplot for Ratings vs. Category
    1. 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.

1. Reviews and Install have some values that are still relatively very high. Before building a linear regression model, you need to reduce the skew. Apply log transformation (np.log1p) to Reviews and Installs.
2. Drop columns App, Last Updated, Current Ver, and Android Ver. These variables are not useful for our task.
3. Get dummy columns for Category, Genres, and Content Rating. This needs to be done as the models do not understand categorical data, and all data should be numeric. Dummy encoding is one way to convert character fields to numeric. Name of dataframe should be **inp2**.

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.

In [1]:
```
#Import the Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

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

Out[2]:

	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	D
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	Des
...	...	...	...	...	...	...	...	...	...	
10836	Sya9a Maroc - FR	FAMILY	4.5	38	53M	5,000+	Free	0	Everyone	
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100+	Free	0	Everyone	
10838	Parkinson Exercices FR	MEDICAL	NaN	3	9.5M	1,000+	Free	0	Everyone	
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	Varies with device	1,000+	Free	0	Mature 17+	
10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307	19M	10,000,000+	Free	0	Everyone	

10841 rows × 13 columns

In [3]:
```
data.head()
```

Loading [MathJax]/extensions/Safe.js

Out[3]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play
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
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity

In [4]:
```
data.describe()
```

Out[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

In [5]:
```
data.shape
```

Out[5]:
```
(10841, 13)
```

In [6]:
```
data.dtypes
```

Loading [MathJax]/extensions/Safe.js

Out[6]:
```
App                object
Category           object
Rating            float64
Reviews            object
Size               object
Installs           object
Type               object
Price              object
Content Rating     object
Genres             object
Last Updated       object
Current Ver        object
Android Ver        object
dtype: object
```

In [7]:
```
data.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
```

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
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
```

In [9]:
```
#Percentage of missing values in col
(data.isnull().sum()/len(data))*100
```

Loading [MathJax]/extensions/Safe.js

Out[9]:
```
App                0.000000
Category           0.000000
Rating            13.596532
Reviews            0.000000
Size               0.000000
Installs           0.000000
Type               0.009224
Price              0.000000
Content Rating     0.009224
Genres             0.000000
Last Updated       0.000000
Current Ver        0.073794
Android Ver        0.027673
dtype: float64
```

In [10]:
```
data.dropna(inplace=True)
```

In [11]:
```
data.isnull().sum()
```

Out[11]:
```
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
```

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

In [12]:
```
def change_size(size):
    if 'M' in size:
        x=size[:-1]
        x=float(x)*1000
        return x
    if 'k' in size:
        x=size[:-1]
        x=float(x)
        return x
    else:
        return None
```

In [13]:
```
data['Size']=data['Size'].apply(change_size)
```

In [14]:
```
data['Size']
```

Loading [MathJax]/extensions/Safe.js

Out[14]:
```
0        19000.0
1        14000.0
2         8700.0
3        25000.0
4         2800.0
          ...
10834     2600.0
10836    53000.0
10837     3600.0
10839        NaN
10840    19000.0
Name: Size, Length: 9360, dtype: float64
```

In [15]:
```
data.isnull().sum()
```

Out[15]:
```
App                0
Category           0
Rating             0
Reviews            0
Size            1637
Installs           0
Type               0
Price              0
Content Rating     0
Genres             0
Last Updated       0
Current Ver        0
Android Ver        0
dtype: int64
```

In [16]:
```
data['Size'].fillna(method='ffill',inplace=True)
```

In [17]:
```
data.isnull().sum()
```

Out[17]:
```
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
```

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

In [18]:
```
data['Reviews']
```

Loading [MathJax]/extensions/Safe.js

Out[18]:
```
0           159
1           967
2         87510
3        215644
4           967
          ...
10834         7
10836        38
10837         4
10839       114
10840    398307
Name: Reviews, Length: 9360, dtype: object
```

In [19]:
```
data.head()
```

Out[19]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genre
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000.0	10,000+	Free	0	Everyone	Art & Desig
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000.0	500,000+	Free	0	Everyone	Art & Design;Preten Pla
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8700.0	5,000,000+	Free	0	Everyone	Art & Desig
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25000.0	50,000,000+	Free	0	Teen	Art & Desig
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2800.0	100,000+	Free	0	Everyone	Art & Design;Creativit

In [20]:
```
data['Reviews'] = data['Reviews'].astype('int')
```

In [21]:
```
data['Reviews']
```

Out[21]:
```
0           159
1           967
2         87510
3        215644
4           967
          ...
10834         7
10836        38
10837         4
10839       114
10840    398307
Name: Reviews, Length: 9360, dtype: int32
```

In [22]:
```
data.dtypes
```

Loading [MathJax]/extensions/Safe.js

Out[22]:
```
App                object
Category           object
Rating            float64
Reviews             int32
Size              float64
Installs           object
Type               object
Price              object
Content Rating     object
Genres             object
Last Updated       object
Current Ver        object
Android Ver        object
dtype: object
```

In [23]:
```
#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
```

In [24]:
```
data['Installs'] = data['Installs'].str.replace('+', '')
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_1864\4230444306.py:1: FutureWarning: The defaul
t value of regex will change from True to False in a future version. In addition, single
character regular expressions will *not* be treated as literal strings when regex=True.
  data['Installs'] = data['Installs'].str.replace('+', '')
```

In [25]:
```
data['Installs']=data['Installs'].str.replace(',','')
```

In [26]:
```
data['Installs']=data['Installs'].astype(int)
```

In [27]:
```
data['Price'] = data['Price'].str.replace('$', '')
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_1864\813483522.py:1: FutureWarning: The default
value of regex will change from True to False in a future version. In addition, single c
haracter regular expressions will *not* be treated as literal strings when regex=True.
  data['Price'] = data['Price'].str.replace('$', '')
```

In [28]:
```
data['Price']=data['Price'].astype(float)
```

1. 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.

In [29]:
```
data[data['Rating']>5]
```

Out[29]:

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver

In [30]:
```
data[data['Rating']<1]
```

Out[30]:

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver

In [31]:
```
#Reviews should not be more than installs as only those who installed can review the app
```

Loading [MathJax]/extensions/Safe.js

In [32]:
```
data[data['Reviews'] > data['Installs']]
```

Out[32]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Curr
2454	KBA-EZ Health Guide	MEDICAL	5.0	4	25000.0	1	Free	0.00	Everyone	Medical	August 2, 2018	1.0
4663	Alarmy (Sleep If U Can) - Pro	LIFESTYLE	4.8	10249	30000.0	10000	Paid	2.49	Everyone	Lifestyle	July 30, 2018	Var v dev
5917	Ra Ga Ba	GAME	5.0	2	20000.0	1	Paid	1.49	Everyone	Arcade	February 8, 2017	1.
6700	Brick Breaker BR	GAME	5.0	7	19000.0	5	Free	0.00	Everyone	Arcade	July 23, 2018	
7402	Trovami se ci riesci	GAME	5.0	11	6100.0	10	Free	0.00	Everyone	Arcade	March 11, 2017	
8591	DN Blog	SOCIAL	5.0	20	4200.0	10	Free	0.00	Teen	Social	July 23, 2018	
10697	Mu.F.O.	GAME	5.0	2	16000.0	1	Paid	0.99	Everyone	Arcade	March 3, 2017	

In [33]:
```
len(data[data['Reviews'] > data['Installs']])
```

Out[33]:
```
7
```

In [34]:
```
#drop these row index
data.drop(data[data['Reviews'] > data['Installs']].index,inplace=True)
```

In [35]:
```
len(data[data['Reviews'] > data['Installs']])
```

Out[35]:
```
0
```

In [36]:
```
data.shape
```

Out[36]:
```
(9353, 13)
```

In [37]:
```
data[(data['Type'] == 'Free') & (data['Price'] > 0)]
```

Out[37]:

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver

1. 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?

Loading [MathJax]/extensions/Safe.js

Histogram for Rating

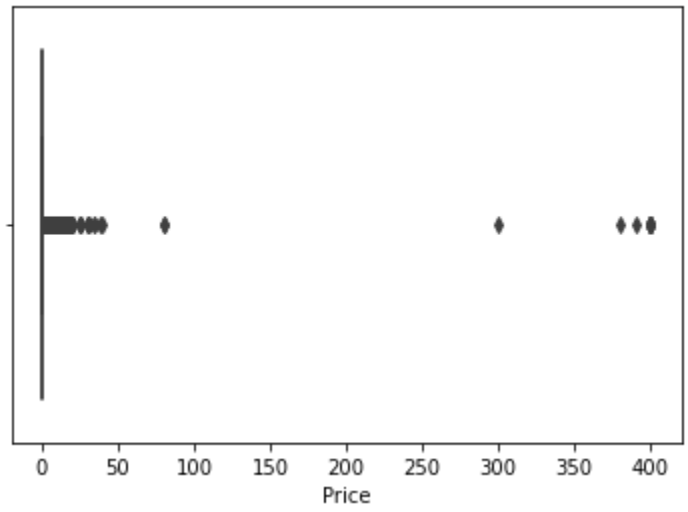
How are the ratings distributed? Is it more toward higher ratings?

Histogram for Size

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

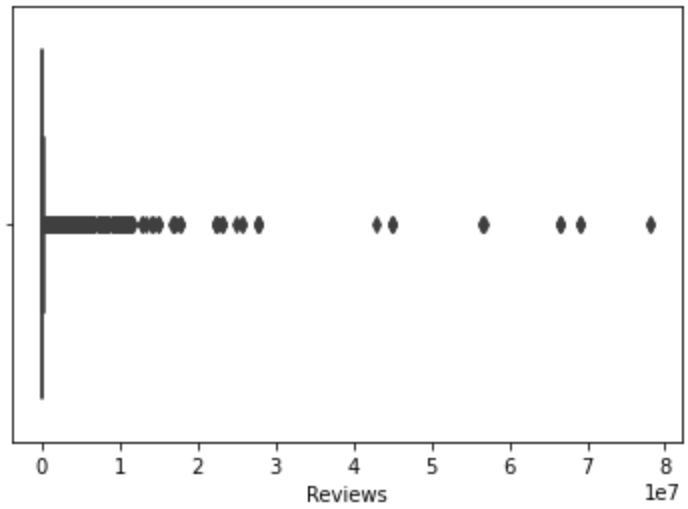
In [38]:
```
sns.boxplot(x='Price',data=data)
#Yes outliers are present in this boxplot from 100 to 400
```

Out[38]:
```
<AxesSubplot:xlabel='Price'>
```



In [39]:
```
sns.boxplot(x='Reviews',data=data)
#yes the boxplot has outliers in the range of 1 lakhs to 8 lakhs
```

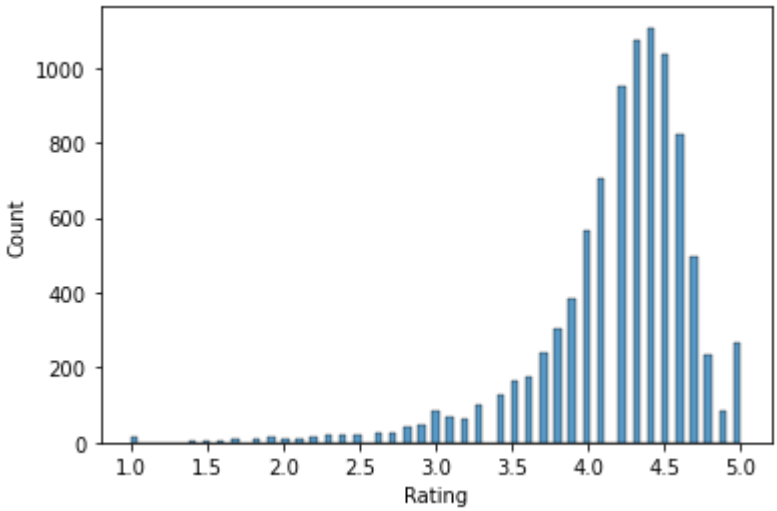
Out[39]:
```
<AxesSubplot:xlabel='Reviews'>
```



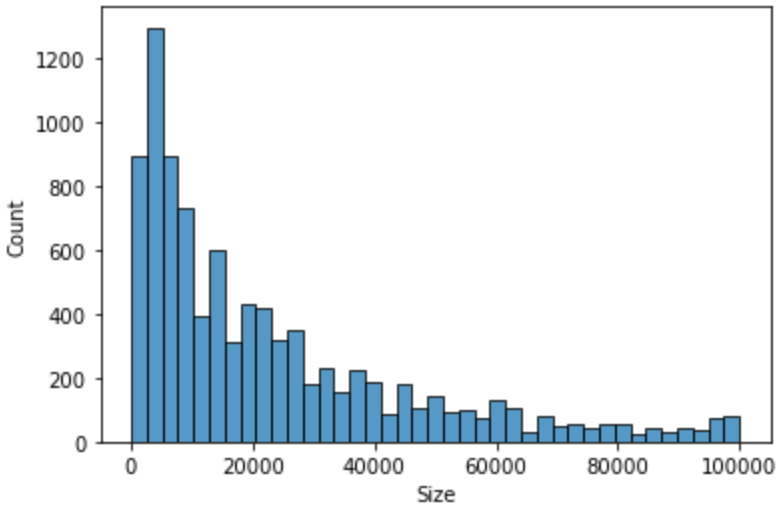
In [40]:
```
sns.histplot(x='Rating',data=data)
#Data is left skewed and ratings are increasing after 2.5 till 5
```

Out[40]:
```
<AxesSubplot:xlabel='Rating', ylabel='Count'>
```

Loading [MathJax]/extensions/Safe.js



In [41]:
```
sns.histplot(x='Size',data=data)
plt.show()
#This is right skewed data and the range of size is from (0,20000)
```



1. 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

Loading [MathJax]/extensions/Safe.js

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

In [42]:
```
data[data['Price']>200]
```

Out[42]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Upd
4197	most expensive app (H)	FAMILY	4.3	6	1500.0	100	Paid	399.99	Everyone	Entertainment	Ju
4362	I'm rich	LIFESTYLE	3.8	718	26000.0	10000	Paid	399.99	Everyone	Lifestyle	Marc
4367	I'm Rich - Trump Edition	LIFESTYLE	3.6	275	7300.0	10000	Paid	400.00	Everyone	Lifestyle	M
5351	I am rich	LIFESTYLE	3.8	3547	1800.0	100000	Paid	399.99	Everyone	Lifestyle	Ja 12,
5354	I am Rich Plus	FAMILY	4.0	856	8700.0	10000	Paid	399.99	Everyone	Entertainment	Ma
5355	I am rich VIP	LIFESTYLE	3.8	411	2600.0	10000	Paid	299.99	Everyone	Lifestyle	Ju
5356	I Am Rich Premium	FINANCE	4.1	1867	4700.0	50000	Paid	399.99	Everyone	Finance	Nove 12,
5357	I am extremely Rich	LIFESTYLE	2.9	41	2900.0	1000	Paid	379.99	Everyone	Lifestyle	J
5358	I am Rich!	FINANCE	3.8	93	22000.0	1000	Paid	399.99	Everyone	Finance	Dece 11,
5359	I am rich(premium)	FINANCE	3.5	472	965.0	5000	Paid	399.99	Everyone	Finance	M
5362	I Am Rich Pro	FAMILY	4.4	201	2700.0	5000	Paid	399.99	Everyone	Entertainment	Ma
5364	I am rich (Most expensive app)	FINANCE	4.1	129	2700.0	1000	Paid	399.99	Teen	Finance	Dece 6,
5366	I Am Rich	FAMILY	3.6	217	4900.0	10000	Paid	389.99	Everyone	Entertainment	Jun
5369	I am Rich	FINANCE	4.3	180	3800.0	5000	Paid	399.99	Everyone	Finance	Marc
5373	I AM RICH PRO PLUS	FINANCE	4.0	36	41000.0	1000	Paid	399.99	Everyone	Finance	Jun

In [43]:
```
len(data[data['Price']>200])
```

Out[43]:
```
15
```

Loading [MathJax]/extensions/Safe.js

In [44]:
```
data[data['Price']>200].size
```

Out[44]:
```
195
```

In [45]:
```
data.shape
```

Out[45]:
```
(9353, 13)
```

In [46]:
```
#Drop the rows whose price is more than 200
data.drop(data[data['Price']>200].index,inplace=True)
```

In [47]:
```
data.shape
```

Out[47]:
```
(9338, 13)
```

In [48]:
```
data[data['Reviews']>=2000000]
```

Out[48]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
139	Wattpad Free Books	BOOKS_AND_REFERENCE	4.6	2914724	3100.0	100000000	Free	0.0	Teen	
335	Messenger – Text and Video Chat for Free	COMMUNICATION	4.0	56642847	35000.0	1000000000	Free	0.0	Everyone	C
336	WhatsApp Messenger	COMMUNICATION	4.4	69119316	35000.0	1000000000	Free	0.0	Everyone	C
338	Google Chrome: Fast & Secure	COMMUNICATION	4.3	9642995	17000.0	1000000000	Free	0.0	Everyone	C
340	Gmail	COMMUNICATION	4.3	4604324	17000.0	1000000000	Free	0.0	Everyone	C
...	...	...	...	...	...	...	...	...	...	...
9166	Modern Combat 5: eSports FPS	GAME	4.3	2903386	58000.0	100000000	Free	0.0	Mature 17+	
9841	Google Earth	TRAVEL_AND_LOCAL	4.3	2339098	63000.0	100000000	Free	0.0	Everyone	
10186	Farm Heroes Saga	FAMILY	4.4	7615646	71000.0	100000000	Free	0.0	Everyone	
10190	Fallout Shelter	FAMILY	4.6	2721923	25000.0	10000000	Free	0.0	Teen	
10327	Garena Free Fire	GAME	4.5	5534114	53000.0	100000000	Free	0.0	Teen	

453 rows × 13 columns

In [49]:
```
len(data[data['Reviews']>2000000])
```

Out[49]:
```
453
```

Loading [MathJax]/extensions/Safe.js

In [50]:
```
data.drop(data[data['Reviews']>2000000].index,inplace=True)
```

In [51]:
```
len(data[data['Reviews']>2000000])
```

Out[51]:
```
0
```

In [52]:
```
data.shape
```

Out[52]:
```
(8885, 13)
```

In [53]:
```
data['Installs'].quantile([0.1,0.25,0.5,0.75,0.9,0.95,0.99])
```

Out[53]:
```
0.10         1000.0
0.25        10000.0
0.50       500000.0
0.75      5000000.0
0.90     10000000.0
0.95     10000000.0
0.99    100000000.0
Name: Installs, dtype: float64
```

In [54]:
```
data[data['Installs']>100000000.0].count()
```

Out[54]:
```
App               20
Category          20
Rating            20
Reviews           20
Size              20
Installs          20
Type              20
Price             20
Content Rating    20
Genres            20
Last Updated      20
Current Ver       20
Android Ver       20
dtype: int64
```

In [55]:
```
data[data['Installs']>100000000.0].shape
```

Out[55]:
```
(20, 13)
```

In [56]:
```
data.drop(data[data['Installs']>=100000000.0].index,inplace=True)
```

In [57]:
```
data.shape
```

Out[57]:
```
(8743, 13)
```

In [58]:
```
#Decide a threshold as cutoff for outlier and drop records having values more than that
#So we have decided that the cutoff threshold for outliers will be 0.99 percentile for i
```

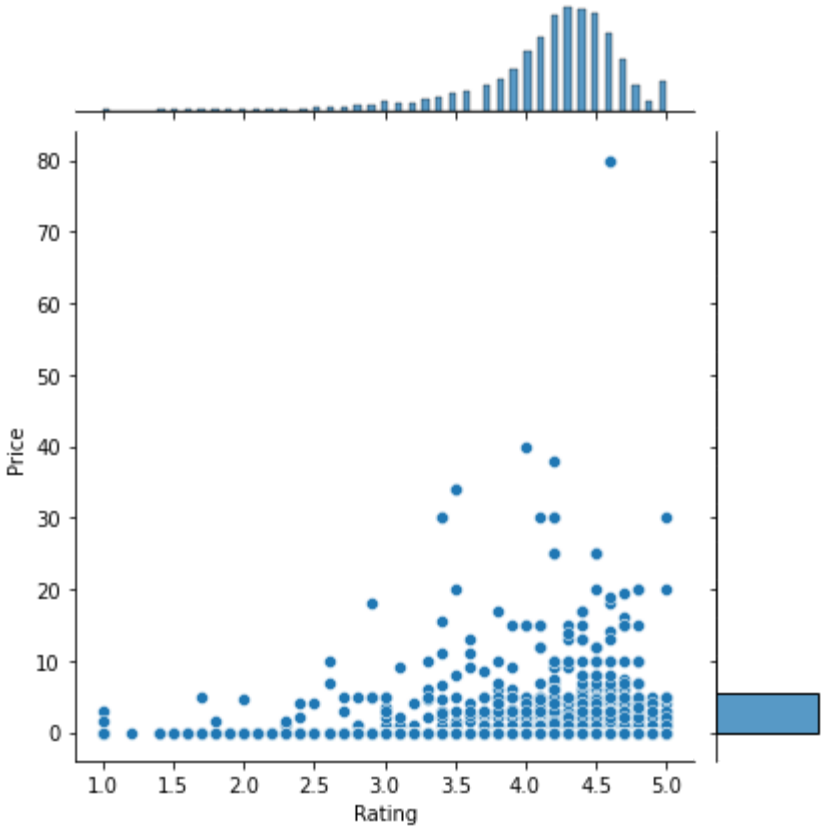
In [59]:
```
#7. Bivariate analysis: Let’s look at how the available predictors relate to the variabl

#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?

 plot/joinplot for Rating vs. Reviews
```

Loading [MathJax]/extensions/Safe.js

```
#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.
```

In [60]:
```
#1.Make scatter plot/joinplot for Rating vs. Price
sns.jointplot(x='Rating',y='Price',data=data)
#Observation which can be drawn from this plot is as the ratings increases price will al
```

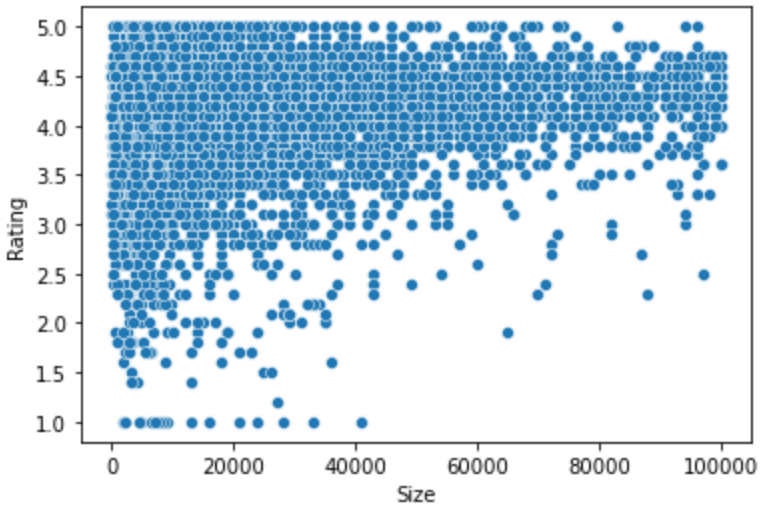
Out[60]:
```
<seaborn.axisgrid.JointGrid at 0x27c68d50610>
```



In [61]:
```
#2.Make scatter plot/joinplot for Rating vs. Size

#1.Are heavier apps rated better?
sns.scatterplot(x='Size',y='Rating',data=data)
#While observing the graph we have seen that as the size of apps increases ratings also
#Yes heavier apps are rated better
```

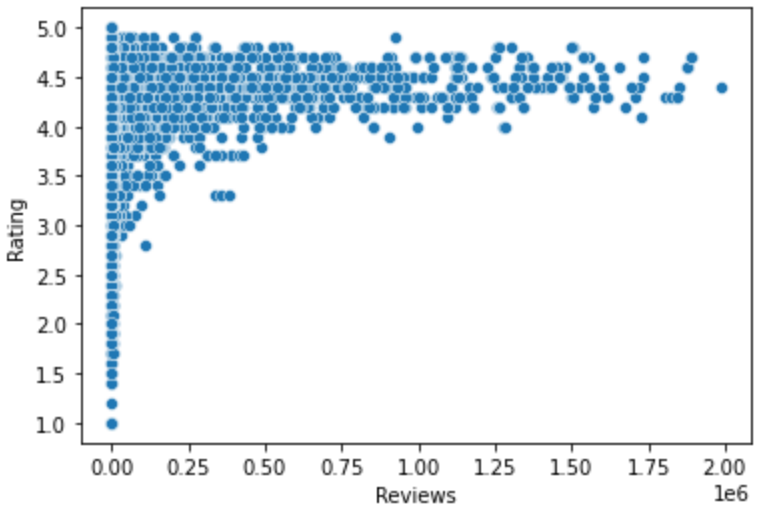
Out[61]:
```
<AxesSubplot:xlabel='Size', ylabel='Rating'>
```

Loading [MathJax]/extensions/Safe.js



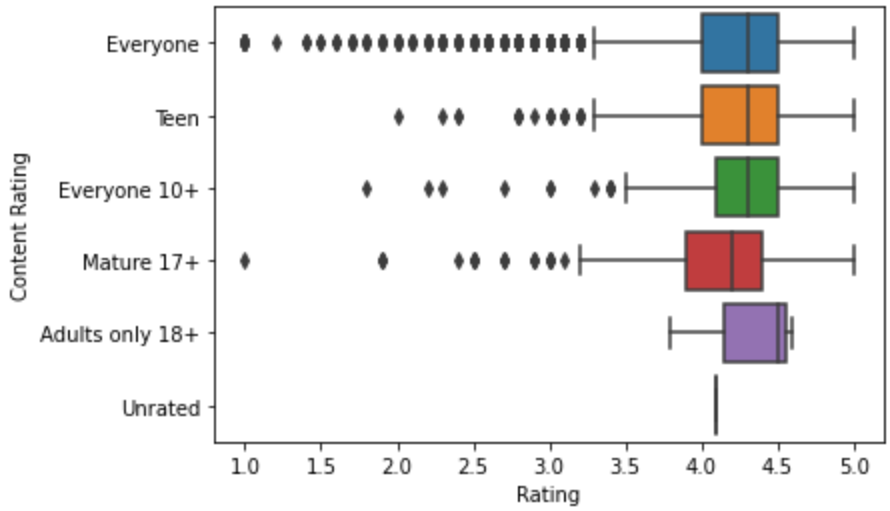
In [62]:
```
#3.Make scatter plot/joinplot for Rating vs. Reviews

#1.Does more review mean a better rating always?
sns.scatterplot(x='Reviews',y='Rating',data=data)
#No more review does not mean better rating always
```

Out[62]:
```
<AxesSubplot:xlabel='Reviews', ylabel='Rating'>
```

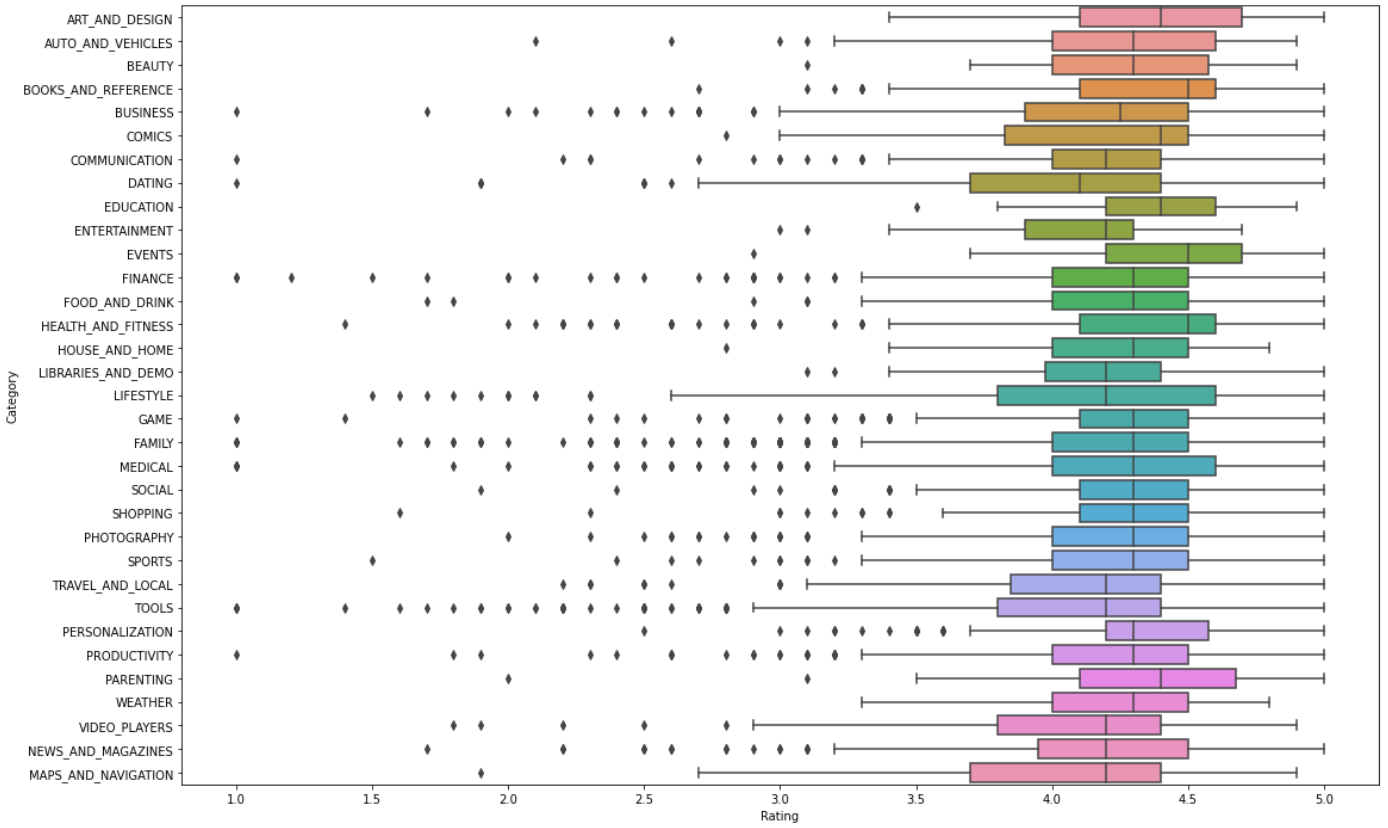


In [63]:
```
#4.Make boxplot for Rating vs. Content Rating

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

sns.boxplot(x='Rating',y='Content Rating',data=data)
plt.show()
#Teen and Everyone are been liked better
```

Loading [MathJax]/extensions/Safe.js



In [64]:
```
plt.figure(figsize=(18,12))
sns.boxplot(x='Rating',y='Category',data=data)
plt.show()
#The genre which has the best ratings are Arts&Design,Education,Events,Health&Fitness,Li
```



In [65]:
```
#9. Train test split  and apply 70-30 split. Name the new dataframes df_train and df_tes

#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.
```

In [66]:
```
data.head()
```

Loading [MathJax]/extensions/Safe.js

Out[66]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000.0	10000	Free	0.0	Everyone	Art & Design
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000.0	500000	Free	0.0	Everyone	Art & Design;Pretend Play
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8700.0	5000000	Free	0.0	Everyone	Art & Design
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25000.0	50000000	Free	0.0	Teen	Art & Design
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2800.0	100000	Free	0.0	Everyone	Art & Design;Creativity

In [67]:
```
inp1=data.copy()
```

In [68]:
```
#8. Data preprocessing

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

#1.Reviews and Install have some values that are still relatively very high. Before buil
```

In [69]:
```
inp1.columns
```

Out[69]:
```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type',
       'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver',
       'Android Ver'],
      dtype='object')
```

In [70]:
```
inp1['Installs']=inp1['Installs'].apply(np.log1p)
```

In [71]:
```
inp1['Reviews']=inp1['Reviews'].apply(np.log1p)
```

In [72]:
```
#2.Drop columns App, Last Updated, Current Ver, and Android Ver. These variables are not
```

In [73]:
```
inp1.drop(['App','Last Updated', 'Current Ver','Android Ver'],axis=1,inplace=True)
```

In [74]:
```
inp1.shape
```

Out[74]:
```
(8743, 9)
```

In [75]:
```
inp1
```

Loading [MathJax]/extensions/Safe.js

Out[75]:

	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genre
0	ART_AND_DESIGN	4.1	5.075174	19000.0	9.210440	Free	0.0	Everyone	Art & Desig
1	ART_AND_DESIGN	3.9	6.875232	14000.0	13.122365	Free	0.0	Everyone	Art & Design;Preten Pla
2	ART_AND_DESIGN	4.7	11.379520	8700.0	15.424949	Free	0.0	Everyone	Art & Desig
3	ART_AND_DESIGN	4.5	12.281389	25000.0	17.727534	Free	0.0	Teen	Art & Desig
4	ART_AND_DESIGN	4.3	6.875232	2800.0	11.512935	Free	0.0	Everyone	Art & Design;Creativit
...	...	...	...	...	...	...	...	...	.
10834	FAMILY	4.0	2.079442	2600.0	6.216606	Free	0.0	Everyone	Educatio
10836	FAMILY	4.5	3.663562	53000.0	8.517393	Free	0.0	Everyone	Educatio
10837	FAMILY	5.0	1.609438	3600.0	4.615121	Free	0.0	Everyone	Educatio
10839	BOOKS_AND_REFERENCE	4.5	4.744932	3600.0	6.908755	Free	0.0	Mature 17+	Books & Referenc
10840	LIFESTYLE	4.5	12.894981	19000.0	16.118096	Free	0.0	Everyone	Lifestyl

8743 rows × 9 columns

In [76]:
```
#3.Get dummy columns for Category, Genres, and Content Rating. This needs to be done as
```

In [77]:
```
#convert data to numeric
#label encoding-blue black brown red yellow-0 1 2 3 4
#dummy encoding convert the data 0 and 1

#dummy variable one hot encoding
```

In [78]:
```
inp2=pd.get_dummies(inp1)
```

In [79]:
```
inp2.shape
```

Out[79]:
```
(8743, 161)
```

In [80]:
```
inp2
```

Loading [MathJax]/extensions/Safe.js

Out[80]:

	Rating	Reviews	Size	Installs	Price	Category_ART_AND_DESIGN	Category_AUTO_AND_VEHICL
0	4.1	5.075174	19000.0	9.210440	0.0	1	
1	3.9	6.875232	14000.0	13.122365	0.0	1	
2	4.7	11.379520	8700.0	15.424949	0.0	1	
3	4.5	12.281389	25000.0	17.727534	0.0	1	
4	4.3	6.875232	2800.0	11.512935	0.0	1	
...	...	...	...	...	...	...	
10834	4.0	2.079442	2600.0	6.216606	0.0	0	
10836	4.5	3.663562	53000.0	8.517393	0.0	0	
10837	5.0	1.609438	3600.0	4.615121	0.0	0	
10839	4.5	4.744932	3600.0	6.908755	0.0	0	
10840	4.5	12.894981	19000.0	16.118096	0.0	0	

8743 rows × 161 columns

In [81]:
```
# Extract features and target
y=inp2.pop('Rating')
X=inp2
```

In [82]:
```
X
```

Out[82]:

	Reviews	Size	Installs	Price	Category_ART_AND_DESIGN	Category_AUTO_AND_VEHICLES	Cat
0	5.075174	19000.0	9.210440	0.0	1	0	
1	6.875232	14000.0	13.122365	0.0	1	0	
2	11.379520	8700.0	15.424949	0.0	1	0	
3	12.281389	25000.0	17.727534	0.0	1	0	
4	6.875232	2800.0	11.512935	0.0	1	0	
...	...	...	...	...	...	...	
10834	2.079442	2600.0	6.216606	0.0	0	0	
10836	3.663562	53000.0	8.517393	0.0	0	0	
10837	1.609438	3600.0	4.615121	0.0	0	0	
10839	4.744932	3600.0	6.908755	0.0	0	0	
10840	12.894981	19000.0	16.118096	0.0	0	0	

8743 rows × 160 columns

In [83]:
```
y
```

Loading [MathJax]/extensions/Safe.js

Out[83]:
```
0        4.1
1        3.9
2        4.7
3        4.5
4        4.3
        ...
10834    4.0
10836    4.5
10837    5.0
10839    4.5
10840    4.5
Name: Rating, Length: 8743, dtype: float64
```

In [84]:
```
#9. Train test split  and apply 70-30 split. Name the new dataframes df_train and df_tes
```

In [85]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=100)
```

In [86]:
```
X_train
```

Out[86]:

	Reviews	Size	Installs	Price	Category_ART_AND_DESIGN	Category_AUTO_AND_VEHICLES	Cat
5705	5.147494	4800.0	9.210440	0.00	0	0	
2981	10.593605	6100.0	13.815512	0.00	0	0	
8381	3.784190	34.0	6.908755	0.00	0	0	
10045	2.397895	11000.0	8.517393	0.00	0	0	
1822	10.130424	82000.0	13.815512	0.00	0	0	
...	...	...	...	...	...	...	
399	12.766131	8300.0	16.118096	0.00	0	0	
81	9.627009	37000.0	13.815512	0.00	0	1	
9869	3.688879	32000.0	6.908755	2.56	0	0	
8516	6.458338	7200.0	11.512935	0.00	0	0	
6791	4.219508	2700.0	9.210440	0.00	0	0	

6120 rows × 160 columns

In [87]:
```
X_train.shape
```

Out[87]:
```
(6120, 160)
```

In [88]:
```
X_test.shape
```

Out[88]:
```
(2623, 160)
```

In [89]:
```
y_test.shape
```

Out[89]:
```
(2623,)
```

In [90]:
```
y_train.shape
```

Out[90]:
```
(6120,)
```

Loading [MathJax]/extensions/Safe.js

In [91]:
```
#Apply Linear Regression
from sklearn.linear_model import LinearRegression #class
linear_reg=LinearRegression()
```

In [92]:
```
linear_reg.fit(X_train,y_train) #object is learning LR on training data
```

Out[92]:
```
LinearRegression()
```

In [93]:
```
#Prediction
y_pred=linear_reg.predict(X_test)
```

In [94]:
```
y_test #Predicted
```

Out[94]:
```
313      4.1
7907     3.8
9825     4.2
5661     3.7
10048    4.2
        ...
7924     4.3
5916     3.1
9071     4.0
10301    5.0
10148    4.2
Name: Rating, Length: 2623, dtype: float64
```

In [95]:
```
y_pred #actual outcome of test
```

Out[95]:
```
array([4.04742694, 3.9705283 , 4.20214468, ..., 4.30969141, 4.29988316,
       4.09565392])
```

In [96]:
```
#Print the error
from sklearn.metrics import mean_squared_error
print("MSE=",mean_squared_error(y_test,y_pred))
```

```
MSE= 0.22901238773854582
```

In [97]:
```
#Print the error
from sklearn.metrics import mean_squared_error
print("MSE=",np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
MSE= 0.4785523876636139
```

In [98]:
```
#print r2 score
from sklearn.metrics import r2_score
print("R2 score=",r2_score(y_test,y_pred))
```

```
R2 score= 0.1533588969461951
```

In [99]:
```
#save the model
import joblib
joblib.dump(linear_reg,'Linear_Regression.sav')
```

Out[99]:
```
['Linear_Regression.sav']
```

In [100…
```
#Load the model
model=joblib.load('Linear_Regression.sav')
model
```

Out[100]:
```
 LinearRegression()
```

In [ ]:

Loading [MathJax]/extensions/Safe.js