RatingProject

January 17, 2023

[1]:
```
import numpy as np                                    #Calling different Libraries of␣
 ↪Python to work on the google playstore project
import pandas as pd
import matplotlib.pyplot as plt,seaborn as sns
%matplotlib inline
import warnings
```

[2]:
```
inp0 = pd.read_csv("googleplaystore.csv",sep=',')     #importing file␣
 ↪googleplaystore through pandas
```

[3]:
```
inp0.head(2)
```

[3]:
```
                                                 App        Category  Rating  \
0  Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN     4.1
1                             Coloring book moana  ART_AND_DESIGN     3.9

  Reviews Size  Installs  Type Price Content Rating  \
0     159  19M   10,000+  Free     0       Everyone
1     967  14M  500,000+  Free     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
```

[4]:
```
inp0.info()              #calling info and shape funtion to know about the type␣
 ↪and shape of entries(num of rows and columns)
print(inp0.shape)
```

```
<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
(10841, 13)
```

[5]:
```
inp0.isnull().sum()                                    #checking for cells with null␣
 ↪values
```

[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
```

[6]:
```
inp0=inp0.dropna(how="any")                    # dropping any cell with null␣
 ↪values and storing it in dataframe inp1
print (inp0.isnull().sum())                    # checking again for null values␣
 ↪in new datatframe
print("After Dropping the shape",inp0.shape)   # checking the shape of new data␣
 ↪set after drooping the null values
inp0.head(2)                                   # around 14% of data which has␣
 ↪null values is lost so proceeding with drop
```

```
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
After Dropping the shape (9360, 13)
```

[6]:
```
                                                 App        Category  Rating  \
0  Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN     4.1
1                             Coloring book moana  ART_AND_DESIGN     3.9

  Reviews Size  Installs  Type Price Content Rating  \
0     159  19M   10,000+  Free     0       Everyone
1     967  14M  500,000+  Free     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
```

0.1 Defining Functions to clean data further

[7]:
```
def clean_price(x):                                #function created for cleaning the␣
↪$ sign out of price and changing it to float
    if x =="0":
        return 0
    else:
        return float(x[1:])
```

[8]:
```
def clean_installs(y):                                # function created for␣
↪removing the , and + sign from installs
    return int(y.replace(',','').replace('+',''))
```

[9]:
```
def change_size(z):
    if 'M' in z:                                        #writing a function to␣
↪change size of apps from Mb to KB
        x = z[:-1]
        x = float (x)*1000
        return x

    elif 'k'in z[-1]:
        x = z[:-1]
        x = float (x)
```

```
        return x
    else:
        return None
```

0.2 Calling Functions to clean Data

[10]:
```
inp0['Installs'] = inp0.Installs.map(clean_installs)
```

[11]:
```
inp0['Price'] = inp0.Price.map(clean_price)
```

[12]:
```
inp0['Size'] = inp0.Size.map(change_size)
```

[13]:
```
inp0['Reviews']=inp0.Reviews.map(int)
```

[14]:
```
inp0.Size.fillna(method='ffill',inplace=True)
```

[15]:
```
inp0.dtypes
```

[15]:
```
App                object
Category           object
Rating            float64
Reviews             int64
Size              float64
Installs            int64
Type               object
Price             float64
Content Rating     object
Genres             object
Last Updated       object
Current Ver        object
Android Ver        object
dtype: object
```

0.3 Doing Sanity Check

[16]:
```
inp0.Rating.describe()      #Rating seems ok without any fault
```

[16]:
```
count    9360.000000
mean        4.191838
std         0.515263
min         1.000000
25%         4.000000
```

```
50%         4.300000
75%         4.500000
max         5.000000
Name: Rating, dtype: float64
```

[17]:
```
inp0.drop(inp0.loc[inp0.Reviews>inp0.Installs].index,axis=0,inplace=True)
↪#Droping rows where reviews are more than installs
```

[18]:
```
len(inp0[(inp0.Type=="Free") & (inp0.Price>0)])          #Checking if any
↪apps mentioned as free has price not zero
```

[18]:
```
0
```

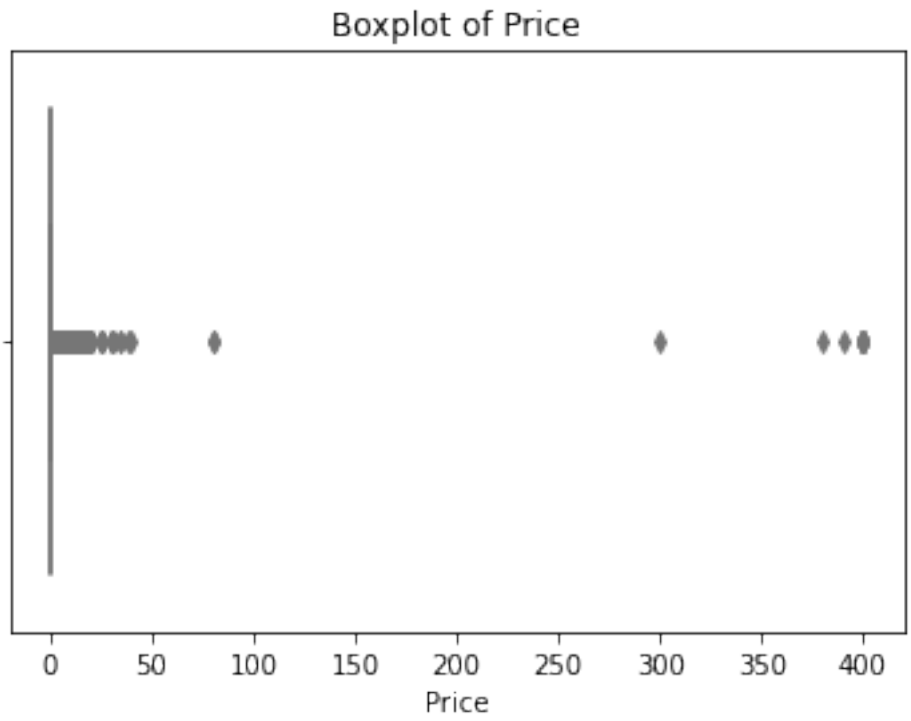
[19]:
```
inp0.shape
```

[19]:
```
(9353, 13)
```

0.4 Performing Univariate analysis

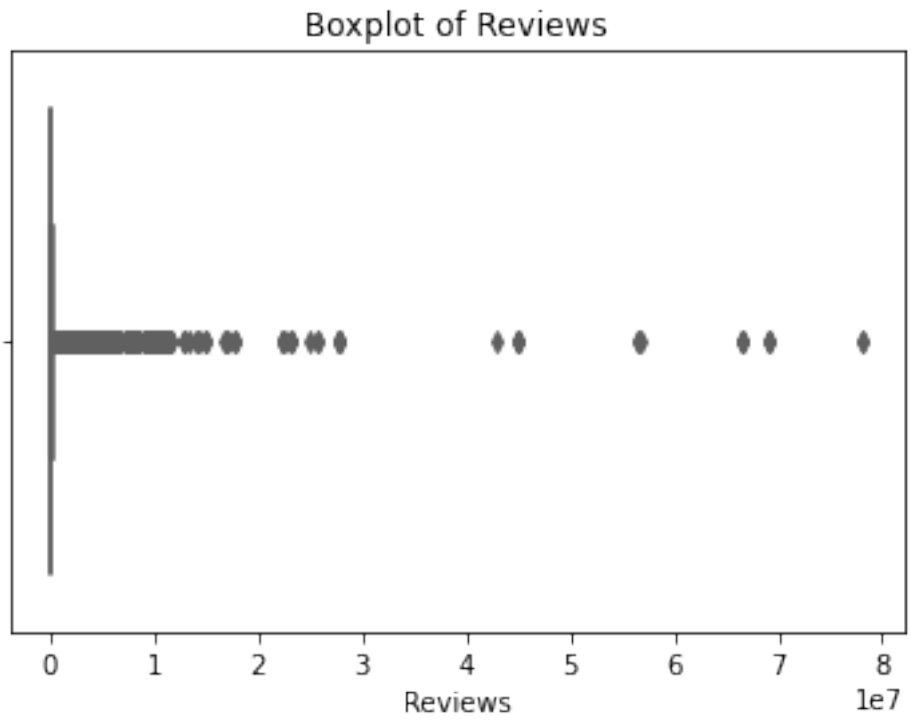
[20]:
```
sns.boxplot(x=inp0.Price,palette="Paired").set(title="Boxplot of Price");
↪#boxplot for Price
inp0.Price.describe()
```

[20]:
```
count    9353.000000
mean        0.961467
std        15.827539
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       400.000000
Name: Price, dtype: float64
```



Most apps are free and Its observed that most of the apps price ranges between $0 to $10 with some having price below $100.
Some apps have exceptionally high price of $300 to $400

[21]:
```
sns.boxplot(x=inp0.Reviews,palette="Blues").set(title="Boxplot of Reviews");
```



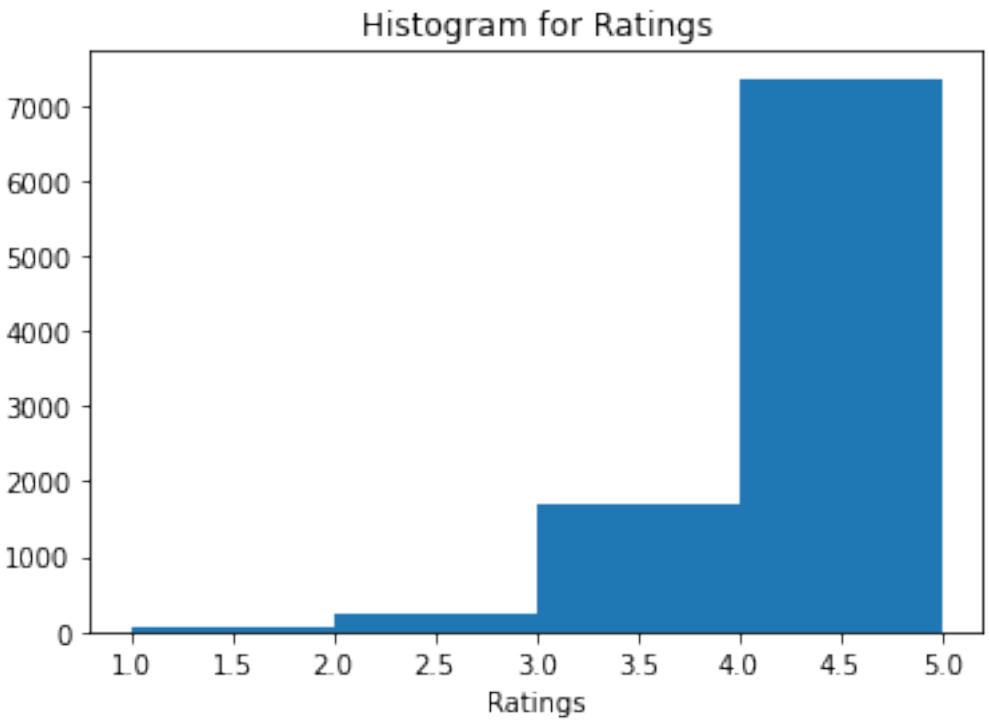
[22]:
```
Q1=inp0['Reviews'].quantile(0.25)
print('Q1',Q1)
Q2=inp0['Reviews'].quantile(0.50)
print('median',Q2)
Q3=inp0['Reviews'].quantile(0.75)
print('upper Quartile',Q3)
maxval=inp0['Reviews'].max()
print('maxvalue',maxval)
```

```
Q1 187.0
median 5967.0
upper Quartile 81747.0
maxvalue 78158306
```

Some apps were found having very high number of Reviews more than 2 million these are most popular apps and can skew the Data

[23]:
```
plt.xlabel('Ratings')
plt.title("Histogram for Ratings")
plt.hist([inp0.Rating],bins=[1,2,3,4,5],rwidth=1)
```

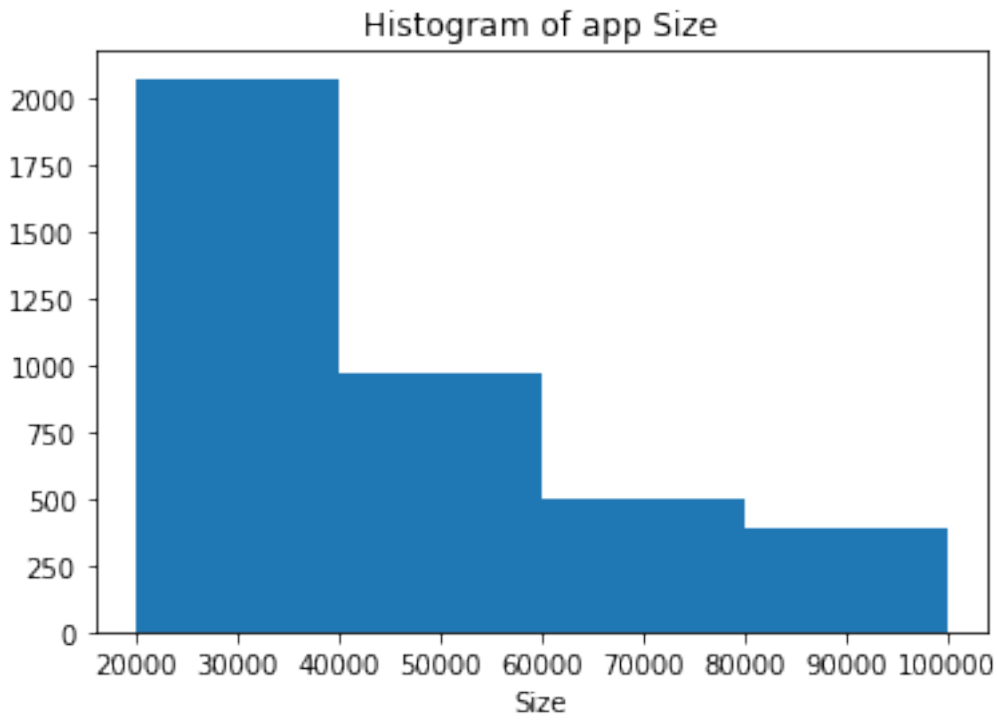
[23]:
```
(array([  56.,  231., 1710., 7356.]),
 array([1, 2, 3, 4, 5]),
 <BarContainer object of 4 artists>)
```



The Data is Skewed towards the left side this means most of the apps have higher ratings between 4 to 5

[24]:
```
plt.xlabel('Size')
plt.title('Histogram of app Size')
plt.hist([inp0.Size],bins=[20000,40000,60000,80000,100000],rwidth=1)
```

[24]:
```
(array([2072.,  965.,  500.,  387.]),
 array([ 20000,  40000,  60000,  80000, 100000]),
 <BarContainer object of 4 artists>)
```



It is Observed that most apps size range between 20 mb upto 60 mb.Some of the apps are larger in size ranging upto 100mb

0.5 Outlier Tretment

[25]:
```
inp0.drop(inp0.loc[inp0.Price>=200].index,axis=0,inplace=True)  #in Boxplot␣
↪analysis noted some apps having high pricce
inp0.Price.describe()                                           # droping apps␣
↪with price above 200
```

[25]:
```
count    9338.000000
mean        0.334412
std         2.170568
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max        79.990000
Name: Price, dtype: float64
```

[26]:
```
inp0.drop(inp0.loc[inp0.Reviews>=2000000].index,axis=0,inplace=True)
```

```
inp0.shape                                                    #in Boxplot analysis␣
↪noted some apps having more than 2Mn reviewsdropping those with more than␣
↪2mn reviews
```

[26]: (8885, 13)

```
[27]: inp0.Installs.quantile([0.1,0.25,0.50,0.70,0.90,0.95,0.99])  #Finding out the␣
      ↪different Percentile Values
```

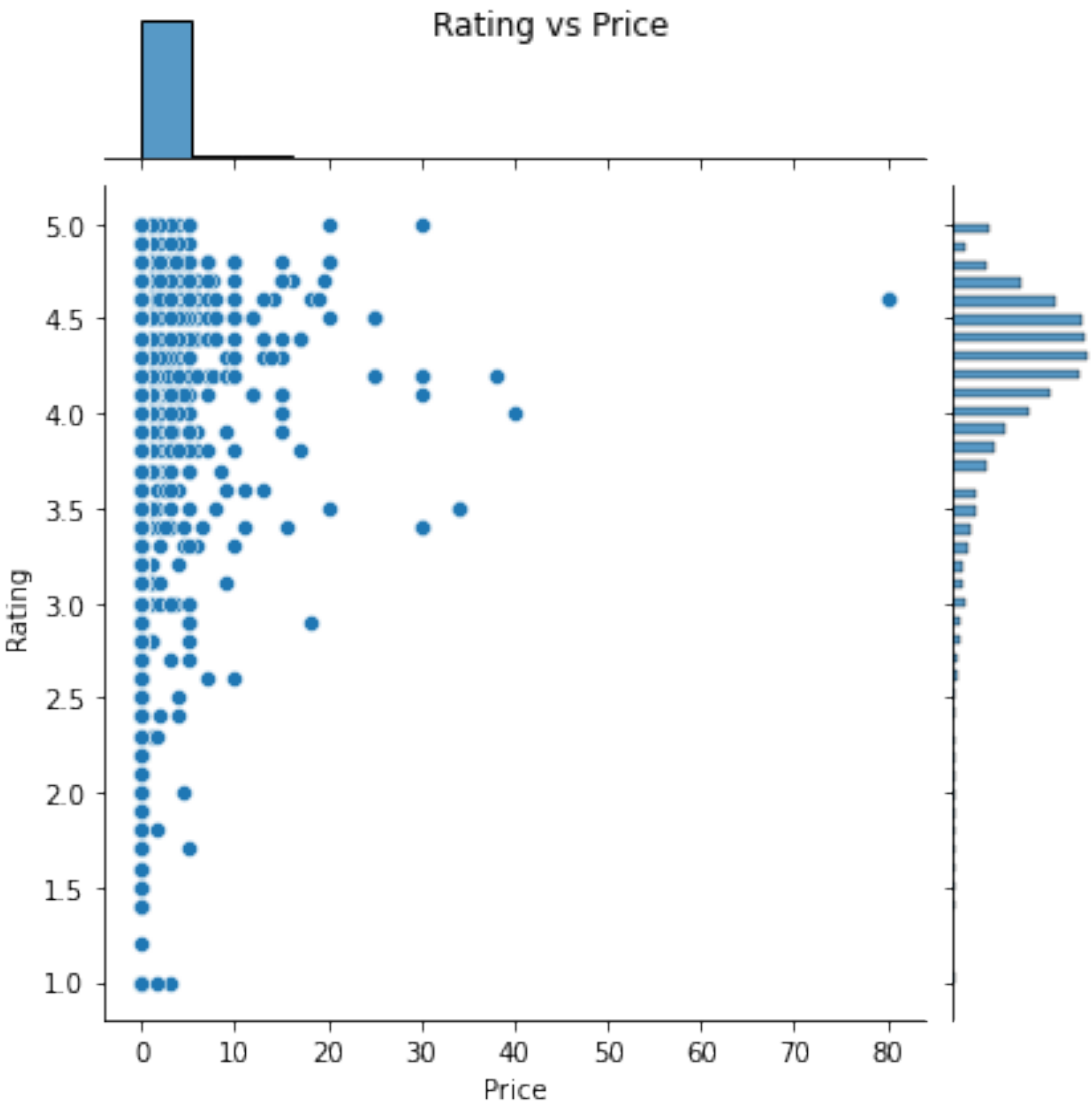
[27]:
```
0.10         1000.0
0.25        10000.0
0.50       500000.0
0.70      1000000.0
0.90     10000000.0
0.95     10000000.0
0.99    100000000.0
Name: Installs, dtype: float64
```

```
[28]: inp0.drop(inp0.loc[inp0.Installs>=100000000].index,axis=0,inplace=True)      ␣
      ↪#Droping Rows with very high number of installs
      inp0.shape
```

[28]: (8743, 13)

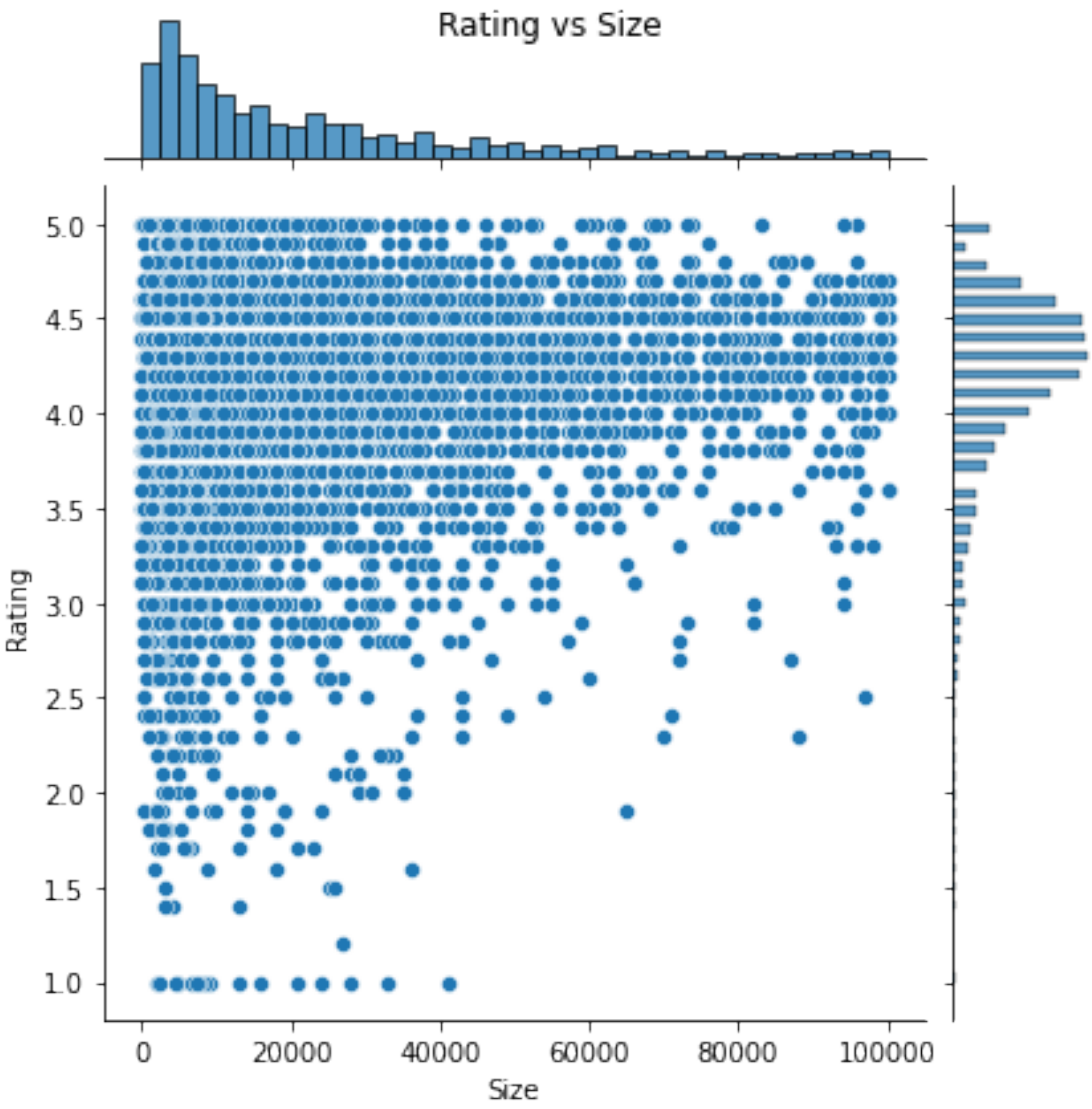
0.6 Bivariate Analysis

```
[29]: plot=sns.jointplot(x=inp0.Price,y=inp0.Rating,kind='scatter');          ␣
      ↪#Checking for affect on Rating with Price
      plot.fig.suptitle("Rating vs Price");
```



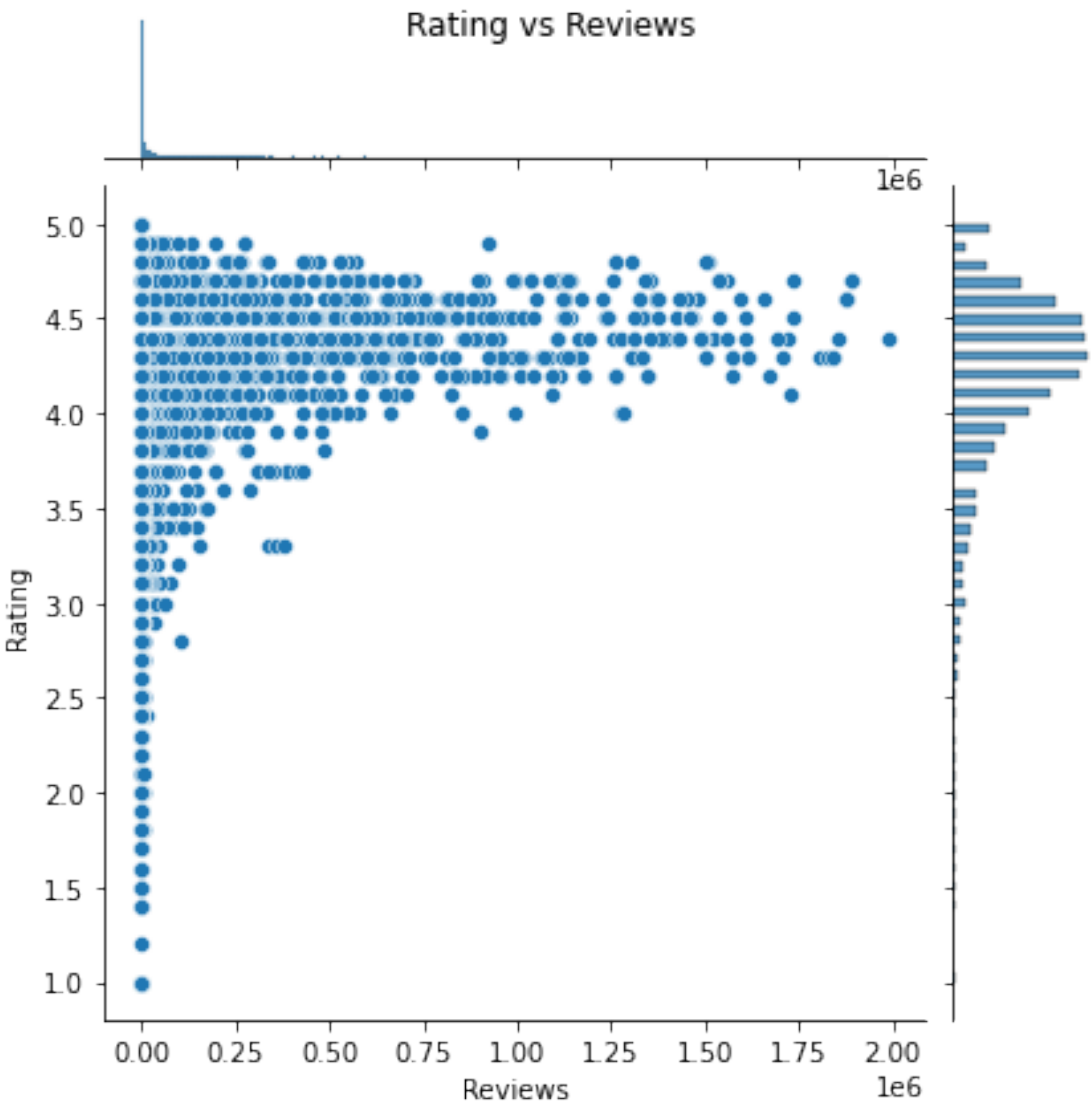
With most of the scatter around 0 we can say that the Rating does not increase with increase in price

[30]:
```
plot=sns.jointplot(x=inp0.Size,y=inp0.Rating,kind='scatter');          #Checking␣
 ↪for affect on Rating for Size
plot.fig.suptitle("Rating vs Size");
```



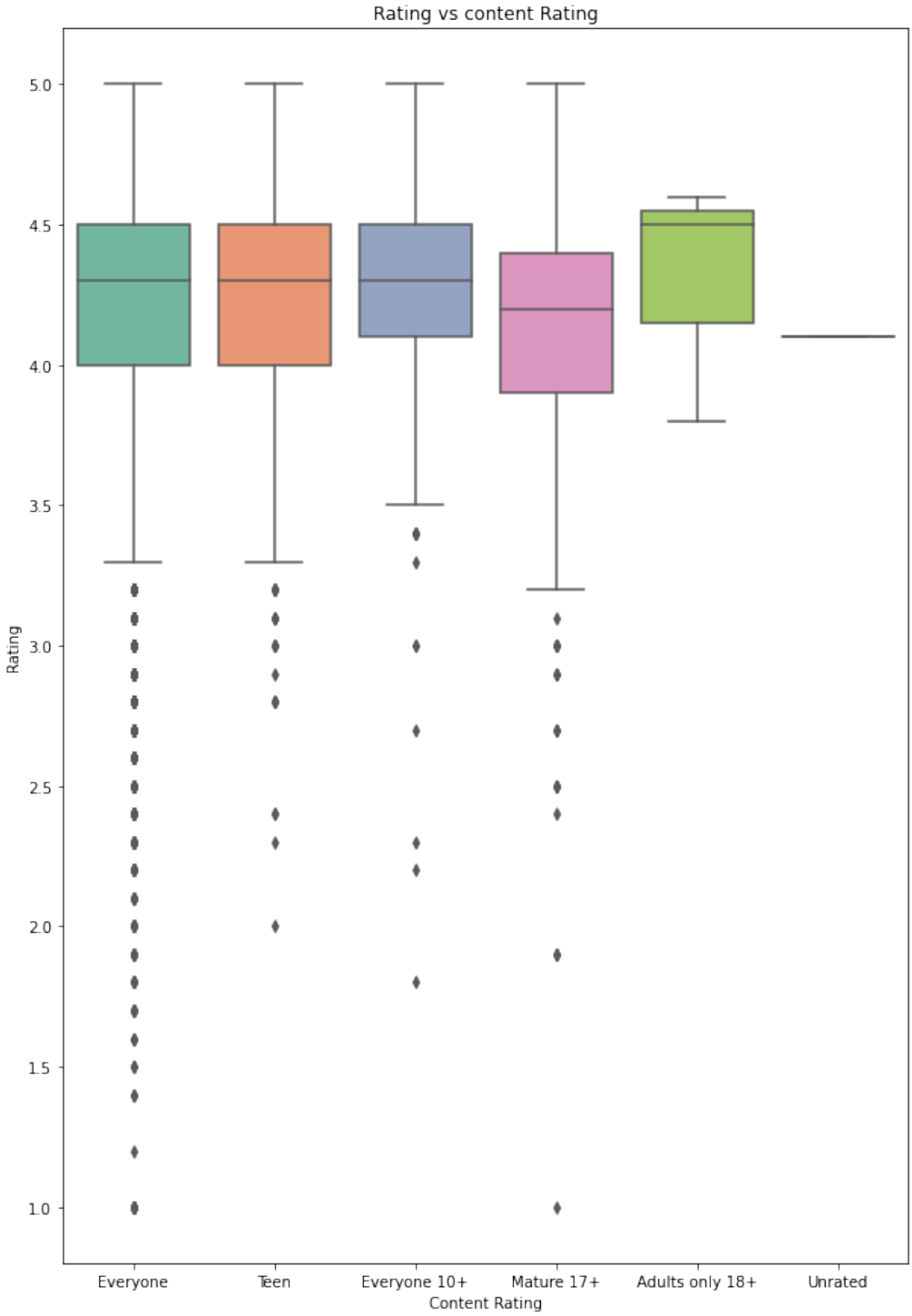
Some of the apps with large size have lower ratings so ratings do not increase with size of app

[31]:
```
plot=sns.jointplot(x=inp0.Reviews,y=inp0.Rating,kind='scatter');
plot.fig.suptitle("Rating vs Reviews");                    #Checking for␣
 ↪effect on Rating for number of Reviews
```

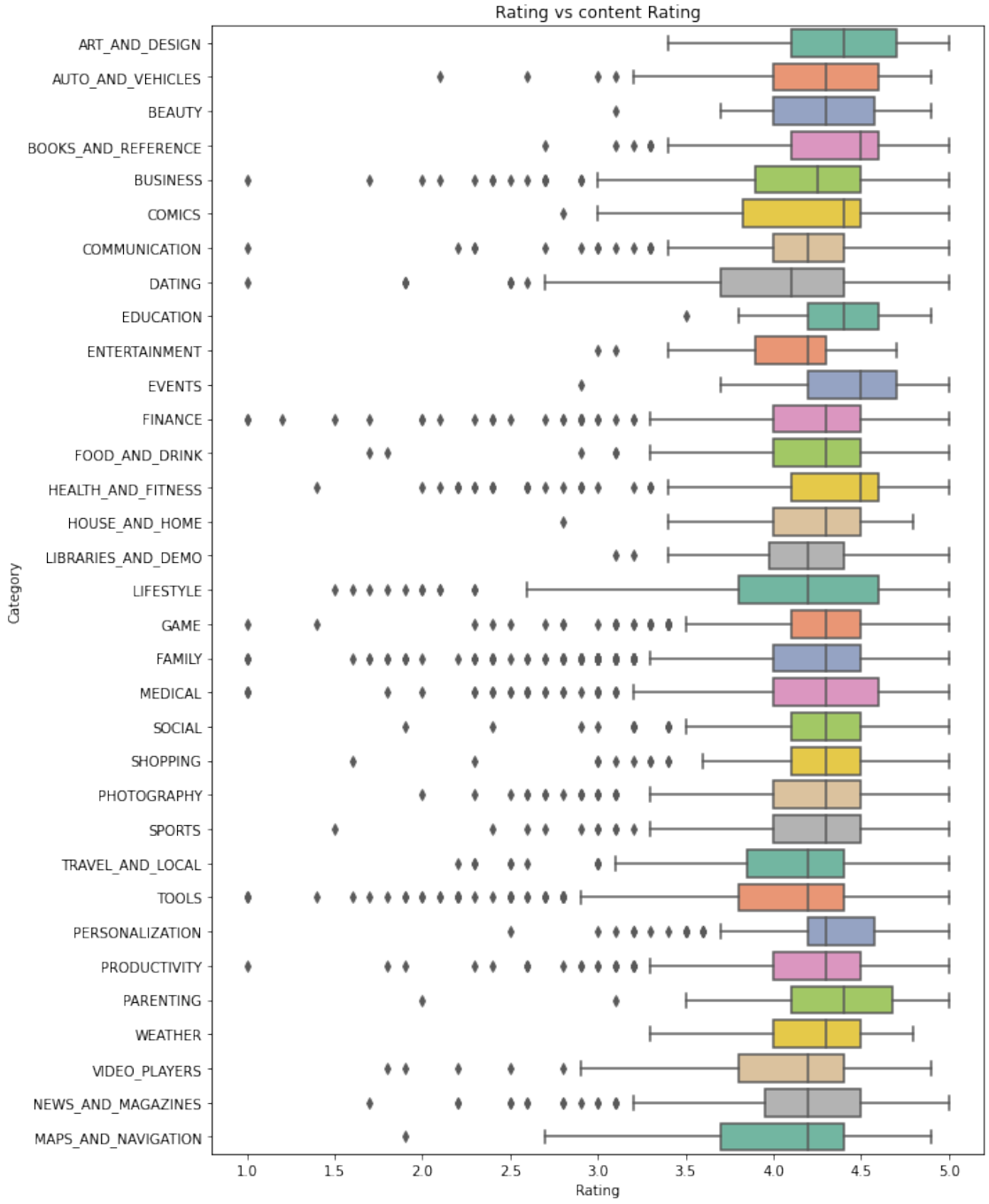


Here it is observed that higher rated apps have more Reviews

[32]:
```
plt.figure(figsize=[10,15])
sns.boxplot(x='Content Rating' ,y='Rating',data=inp0,palette='Set2').
 ↪set(title='Rating vs content Rating');

## Checking the effect on Rating content wise
```

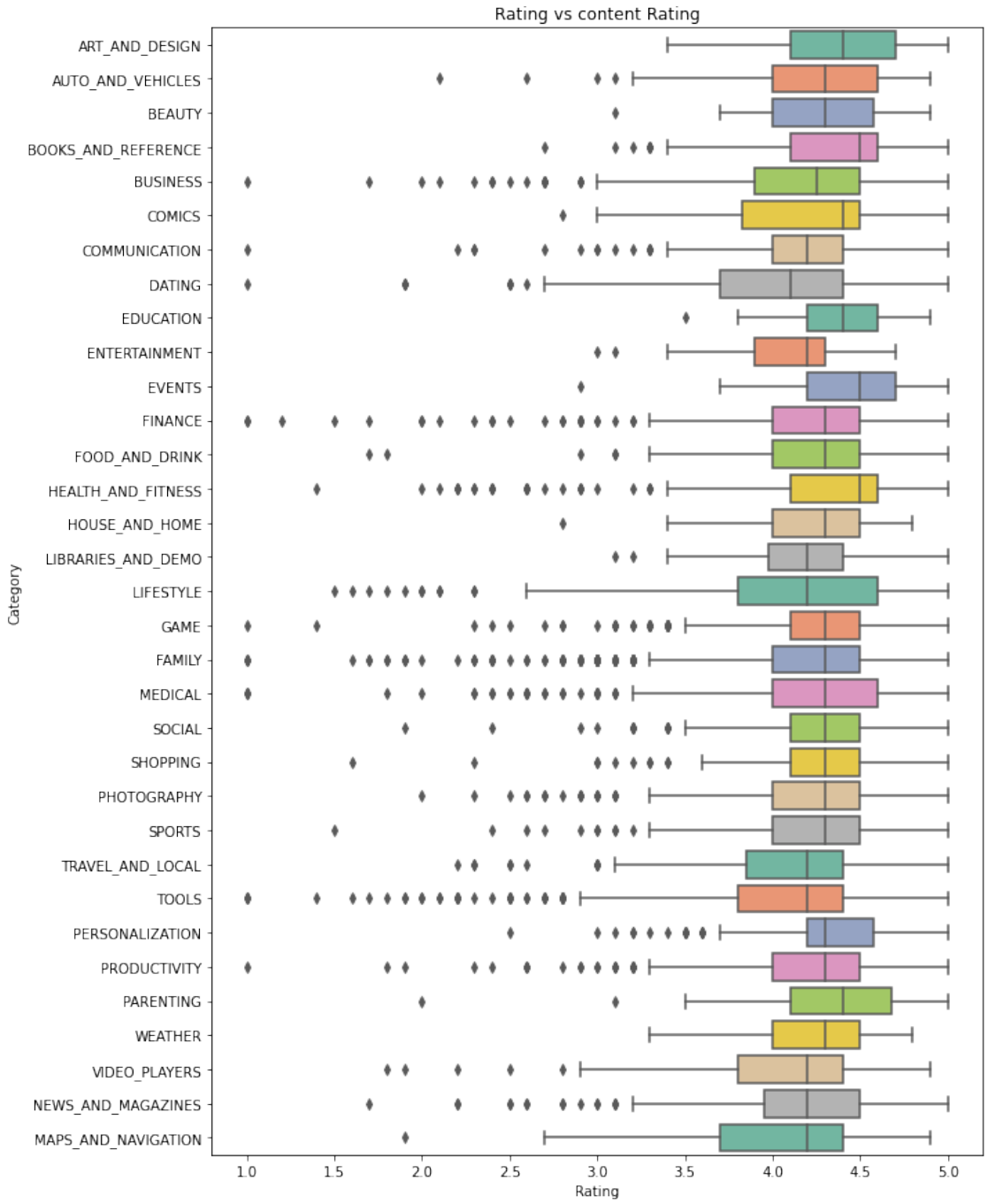

Rating vs content Rating
Rating
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Everyone
Teen
Everyone 10+
Mature 17+
Adults only 18+
Unrated
Content Rating

[33]:
```
plt.figure(figsize=[10,15])
sns.boxplot(y='Category' ,x='Rating',data=inp0,palette='Set2').
 ↪set(title='Rating vs content Rating');

# Checking the effect on Rating for every content
```



It is Observed that the apps available to everyone has maximum outliers so this catogery is most popular and adult only apps have no outliers

[34]:
```
plt.figure(figsize=[10,15])
sns.boxplot(y='Category' ,x='Rating',data=inp0,palette='Set2').
 ↪set(title='Rating vs content Rating');

# Checking the effect on Rating for every content
```



Here it is observed that Event has the highest median rating so we can say Event has one of the best ratings.

0.7 Data Preprocessing

[35]:
```
inp1=inp0.copy()        # creating a copy of data
```

[36]:
```
inp1.Reviews=inp1.Reviews.apply(np.log1p)              #Log transformation␣
↪applied to reduce the skew
inp1.Reviews
```

[36]:
```
0         5.075174
1         6.875232
2        11.379520
3        12.281389
4         6.875232
            …
10834     2.079442
10836     3.663562
10837     1.609438
10839     4.744932
10840    12.894981
Name: Reviews, Length: 8743, dtype: float64
```

[37]:
```
inp1.Installs=inp1.Installs.apply(np.log1p)              #Log transformation␣
↪applied to reduce the skew
inp1.Installs
```

[37]:
```
0         9.210440
1        13.122365
2        15.424949
3        17.727534
4        11.512935
            …
10834     6.216606
10836     8.517393
10837     4.615121
10839     6.908755
10840    16.118096
Name: Installs, Length: 8743, dtype: float64
```

[38]:
```
inp1.columns                              #Getting Name details of columns
```

[38]:
```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type',
       'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver',
       'Android Ver'],
      dtype='object')
```

[39]:
```
inp1.drop(['App','Last Updated','Current Ver','Android␣
↪Ver'],axis=1,inplace=True)   #Droping columns not necessary for us
inp1.shape
```

[39]:
```
(8743, 9)
```

[40]:
```
dummies=pd.get_dummies(inp1.Category)
dummies1=pd.get_dummies(inp1.Genres)
dummies2=pd.get_dummies(inp1['Content Rating'])
inp3=pd.concat([inp1,dummies,dummies1,dummies2],axis=1)
inp4=inp3.drop(["Category","Genres","Content Rating","Type"],axis=1)
inp2=inp4.drop(['ART_AND_DESIGN','Teen','Art & Design'],axis=1)
inp2.columns
```

[40]:
```
Index(['Rating', 'Reviews', 'Size', 'Installs', 'Price', 'AUTO_AND_VEHICLES',
       'BEAUTY', 'BOOKS_AND_REFERENCE', 'BUSINESS', 'COMICS',
       …
       'Video Players & Editors', 'Video Players & Editors;Creativity',
       'Video Players & Editors;Music & Video', 'Weather', 'Word',
       'Adults only 18+', 'Everyone', 'Everyone 10+', 'Mature 17+', 'Unrated'],
      dtype='object', length=156)
```

[41]:
```
df_train=inp2.drop(['Rating'],axis=1)
display(df_train.columns)
df_test=inp2['Rating']
df_test.head()
```

```
Index(['Reviews', 'Size', 'Installs', 'Price', 'AUTO_AND_VEHICLES', 'BEAUTY',
       'BOOKS_AND_REFERENCE', 'BUSINESS', 'COMICS', 'COMMUNICATION',
       ...
       'Video Players & Editors', 'Video Players & Editors;Creativity',
       'Video Players & Editors;Music & Video', 'Weather', 'Word',
       'Adults only 18+', 'Everyone', 'Everyone 10+', 'Mature 17+', 'Unrated'],
      dtype='object', length=155)
```

[41]:
```
0    4.1
1    3.9
2    4.7
3    4.5
4    4.3
Name: Rating, dtype: float64
```

0.8 Train Test Split and applying 70-30 split

[42]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_train,df_test, test_size␣
↪= 0.30, random_state = 100)
display ("X Train Head",X_train.head())
display ("YTrain Head",y_train.head())
display ("X Test Head",X_test.head())
display ("Y Test Head",y_test.head())
```

'X Train Head'

	Reviews	Size	Installs	Price	AUTO_AND_VEHICLES	BEAUTY	\
5705	5.147494	4800.0	9.210440	0.0	0	0	
2981	10.593605	6100.0	13.815512	0.0	0	0	
8381	3.784190	34.0	6.908755	0.0	0	0	
10045	2.397895	11000.0	8.517393	0.0	0	0	
1822	10.130424	82000.0	13.815512	0.0	0	0	

	BOOKS_AND_REFERENCE	BUSINESS	COMICS	COMMUNICATION	...	\
5705	0	0	0	0	...	
2981	0	0	0	0	...	
8381	0	0	0	0	...	
10045	0	0	0	0	...	
1822	0	0	0	0	...	

	Video Players & Editors	Video Players & Editors;Creativity	\
5705	1	0	
2981	0	0	
8381	0	0	
10045	0	0	
1822	0	0	

	Video Players & Editors;Music & Video	Weather	Word	Adults only 18+	\
5705	0	0	0	0	
2981	0	0	0	0	
8381	0	0	0	0	
10045	0	0	0	0	
1822	0	0	0	0	

	Everyone	Everyone 10+	Mature 17+	Unrated
5705	1	0	0	0
2981	1	0	0	0
8381	1	0	0	0
10045	1	0	0	0
1822	0	0	0	0

```
[5 rows x 155 columns]
```

```
'YTrain Head'
```

```
5705     3.3
2981     4.3
8381     4.2
10045    4.1
1822     4.3
Name: Rating, dtype: float64
```

```
'X Test Head'
```

```
         Reviews     Size   Installs  Price  AUTO_AND_VEHICLES  BEAUTY  \
313     9.400051  21000.0  13.815512    0.0                  0       0
7907    6.175867  15000.0  10.819798    0.0                  0       0
9825   10.125871   9200.0  13.815512    0.0                  0       0
5661    6.525030   8200.0   9.210440    0.0                  0       0
10048   7.899524    160.0  11.512935    0.0                  0       0

       BOOKS_AND_REFERENCE  BUSINESS  COMICS  COMMUNICATION  ...  \
313                      0         0       1              0  ...
7907                     0         0       0              0  ...
9825                     0         0       0              0  ...
5661                     0         0       0              0  ...
10048                    0         0       0              0  ...

       Video Players & Editors  Video Players & Editors;Creativity  \
313                          0                                   0
7907                         0                                   0
9825                         0                                   0
5661                         0                                   0
10048                        0                                   0

       Video Players & Editors;Music & Video  Weather  Word  Adults only 18+  \
313                                        0        0     0                 0
7907                                       0        0     0                 0
9825                                       0        0     0                 0
5661                                       0        0     0                 0
10048                                      0        0     0                 0

       Everyone  Everyone 10+  Mature 17+  Unrated
313           0             0           1        0
7907          1             0           0        0
9825          1             0           0        0
5661          1             0           0        0
```

```
10048         1              0           0        0

[5 rows x 155 columns]


'Y Test Head'


313      4.1
7907     3.8
9825     4.2
5661     3.7
10048    4.2
Name: Rating, dtype: float64
```

0.9 Seperating dataframes into x and y test,train frames

[43]:
```
X_train.to_csv("X_train.csv")
X_test.to_csv("X_test.csv")
y_train.to_csv("Y_train.csv")
y_test.to_csv("Y_test.csv")
```

0.10 Model Building

[44]:
```
from sklearn.linear_model import LinearRegression
linear_reg=LinearRegression()
```

[45]:
```
linear_reg.fit(X_train,y_train)                                    #using Linear␣
 ↪regression as a technique
print(round(linear_reg.intercept_,3))
print(np.round(linear_reg.coef_,3))
```

```
4.865
[ 0.167 -0.    -0.144 -0.005 -0.096 -0.018 -0.051 -0.153  0.029 -0.178
 -0.22  -0.004 -0.105  0.044 -0.001 -0.178 -0.127  0.18  -0.132 -0.148
 -0.106 -0.137 -0.201 -0.109 -0.152 -0.196 -0.077 -0.14  -0.153 -0.122
 -0.137  0.062 -0.19  -0.201 -0.037 -0.126 -0.471 -0.263 -0.544 -0.389
 -0.067 -0.516 -0.443 -0.276 -0.03   0.156 -0.215 -0.096 -0.018 -0.48
 -0.41  -0.177 -0.    -0.051 -0.304 -0.153 -0.624 -0.517  0.115 -0.438
 -0.439 -0.336 -0.003 -0.118 -0.109 -0.195 -0.321 -0.4    0.429 -0.178
  0.    -0.22  -0.127  0.085 -0.229  0.159 -0.073 -0.309 -0.026 -0.561
 -0.229  0.161 -0.411 -0.05  -0.274 -0.323 -0.215 -0.119  0.013 -0.
 -0.228 -0.332  0.044 -0.178 -0.127 -0.132 -0.677 -0.024 -0.148 -0.106
 -0.137 -0.245 -0.    -0.201 -0.109 -0.526  0.127 -0.136 -0.152  0.138
 -0.242 -0.182  0.091 -0.077 -0.14  -0.153 -0.179 -0.267 -0.149 -0.17
```

```
 0.281 -0.553 -0.128  0.    -0.381 -0.357  0.    -0.537 -0.122 -0.345
-0.184 -0.336 -0.378 -0.137 -0.348 -0.071 -0.46  -0.23  -0.421  0.267
-0.213  0.023 -0.167 -0.034 -0.722 -0.364 -0.516 -0.42  -0.126 -0.363
 0.044 -0.005 -0.01  -0.017  0.015]
```

[46]:
```
list(zip(df_train,np.round(linear_reg.coef_,3)))          # pairing the column␣
↪names with coeff
```

[46]:
```
[('Reviews', 0.167),
 ('Size', -0.0),
 ('Installs', -0.144),
 ('Price', -0.005),
 ('AUTO_AND_VEHICLES', -0.096),
 ('BEAUTY', -0.018),
 ('BOOKS_AND_REFERENCE', -0.051),
 ('BUSINESS', -0.153),
 ('COMICS', 0.029),
 ('COMMUNICATION', -0.178),
 ('DATING', -0.22),
 ('EDUCATION', -0.004),
 ('ENTERTAINMENT', -0.105),
 ('EVENTS', 0.044),
 ('FAMILY', -0.001),
 ('FINANCE', -0.178),
 ('FOOD_AND_DRINK', -0.127),
 ('GAME', 0.18),
 ('HEALTH_AND_FITNESS', -0.132),
 ('HOUSE_AND_HOME', -0.148),
 ('LIBRARIES_AND_DEMO', -0.106),
 ('LIFESTYLE', -0.137),
 ('MAPS_AND_NAVIGATION', -0.201),
 ('MEDICAL', -0.109),
 ('NEWS_AND_MAGAZINES', -0.152),
 ('PARENTING', -0.196),
 ('PERSONALIZATION', -0.077),
 ('PHOTOGRAPHY', -0.14),
 ('PRODUCTIVITY', -0.153),
 ('SHOPPING', -0.122),
 ('SOCIAL', -0.137),
 ('SPORTS', 0.062),
 ('TOOLS', -0.19),
 ('TRAVEL_AND_LOCAL', -0.201),
 ('VIDEO_PLAYERS', -0.037),
 ('WEATHER', -0.126),
 ('Action', -0.471),
 ('Action;Action & Adventure', -0.263),
 ('Adventure', -0.544),
```

```
('Adventure;Action & Adventure', -0.389),
('Adventure;Brain Games', -0.067),
('Adventure;Education', -0.516),
('Arcade', -0.443),
('Arcade;Action & Adventure', -0.276),
('Arcade;Pretend Play', -0.03),
('Art & Design;Creativity', 0.156),
('Art & Design;Pretend Play', -0.215),
('Auto & Vehicles', -0.096),
('Beauty', -0.018),
('Board', -0.48),
('Board;Action & Adventure', -0.41),
('Board;Brain Games', -0.177),
('Board;Pretend Play', -0.0),
('Books & Reference', -0.051),
('Books & Reference;Education', -0.304),
('Business', -0.153),
('Card', -0.624),
('Card;Action & Adventure', -0.517),
('Card;Brain Games', 0.115),
('Casino', -0.438),
('Casual', -0.439),
('Casual;Action & Adventure', -0.336),
('Casual;Brain Games', -0.003),
('Casual;Creativity', -0.118),
('Casual;Education', -0.109),
('Casual;Music & Video', -0.195),
('Casual;Pretend Play', -0.321),
('Comics', -0.4),
('Comics;Creativity', 0.429),
('Communication', -0.178),
('Communication;Creativity', 0.0),
('Dating', -0.22),
('Education', -0.127),
('Education;Action & Adventure', 0.085),
('Education;Brain Games', -0.229),
('Education;Creativity', 0.159),
('Education;Education', -0.073),
('Education;Music & Video', -0.309),
('Education;Pretend Play', -0.026),
('Educational', -0.561),
('Educational;Action & Adventure', -0.229),
('Educational;Brain Games', 0.161),
('Educational;Creativity', -0.411),
('Educational;Education', -0.05),
('Educational;Pretend Play', -0.274),
('Entertainment', -0.323),
```

```
('Entertainment;Action & Adventure', -0.215),
('Entertainment;Brain Games', -0.119),
('Entertainment;Creativity', 0.013),
('Entertainment;Education', -0.0),
('Entertainment;Music & Video', -0.228),
('Entertainment;Pretend Play', -0.332),
('Events', 0.044),
('Finance', -0.178),
('Food & Drink', -0.127),
('Health & Fitness', -0.132),
('Health & Fitness;Action & Adventure', -0.677),
('Health & Fitness;Education', -0.024),
('House & Home', -0.148),
('Libraries & Demo', -0.106),
('Lifestyle', -0.137),
('Lifestyle;Education', -0.245),
('Lifestyle;Pretend Play', -0.0),
('Maps & Navigation', -0.201),
('Medical', -0.109),
('Music', -0.526),
('Music & Audio;Music & Video', 0.127),
('Music;Music & Video', -0.136),
('News & Magazines', -0.152),
('Parenting', 0.138),
('Parenting;Brain Games', -0.242),
('Parenting;Education', -0.182),
('Parenting;Music & Video', 0.091),
('Personalization', -0.077),
('Photography', -0.14),
('Productivity', -0.153),
('Puzzle', -0.179),
('Puzzle;Action & Adventure', -0.267),
('Puzzle;Brain Games', -0.149),
('Puzzle;Creativity', -0.17),
('Puzzle;Education', 0.281),
('Racing', -0.553),
('Racing;Action & Adventure', -0.128),
('Racing;Pretend Play', 0.0),
('Role Playing', -0.381),
('Role Playing;Action & Adventure', -0.357),
('Role Playing;Brain Games', 0.0),
('Role Playing;Pretend Play', -0.537),
('Shopping', -0.122),
('Simulation', -0.345),
('Simulation;Action & Adventure', -0.184),
('Simulation;Education', -0.336),
('Simulation;Pretend Play', -0.378),
```

```
 ('Social', -0.137),
 ('Sports', -0.348),
 ('Sports;Action & Adventure', -0.071),
 ('Strategy', -0.46),
 ('Strategy;Action & Adventure', -0.23),
 ('Strategy;Creativity', -0.421),
 ('Strategy;Education', 0.267),
 ('Tools', -0.213),
 ('Tools;Education', 0.023),
 ('Travel & Local', -0.167),
 ('Travel & Local;Action & Adventure', -0.034),
 ('Trivia', -0.722),
 ('Video Players & Editors', -0.364),
 ('Video Players & Editors;Creativity', -0.516),
 ('Video Players & Editors;Music & Video', -0.42),
 ('Weather', -0.126),
 ('Word', -0.363),
 ('Adults only 18+', 0.044),
 ('Everyone', -0.005),
 ('Everyone 10+', -0.01),
 ('Mature 17+', -0.017),
 ('Unrated', 0.015)]
```

[47]:
```
y_pred=linear_reg.predict(X_train)                    #Predicting on train␣
 ↪data set
y_pred[:100]
```

[47]:
```
array([3.99107288, 4.35299888, 4.19101491, 3.58987274, 4.34717279,
       4.1545674 , 4.27408456, 3.98981896, 3.91594699, 4.22704836,
       4.53493031, 4.21481064, 3.84983909, 3.72335425, 4.43330816,
       4.03120358, 4.33651022, 4.12274307, 4.48341806, 4.47468941,
       4.41410426, 4.63592465, 4.24658955, 4.19487988, 4.4572849 ,
       3.80495909, 4.2910981 , 4.0512281 , 3.9703881 , 4.43402803,
       4.11819876, 4.57294277, 4.26058595, 4.03544203, 3.91380446,
       4.27911776, 4.18040207, 4.38626911, 4.18562455, 3.926029  ,
       4.32700268, 4.3252202 , 4.25129934, 3.93851768, 4.13229574,
       4.13787897, 4.45888166, 4.20496919, 4.15950128, 4.21519994,
       4.4085249 , 4.49692198, 4.0302539 , 4.0619659 , 4.23769852,
       4.08624589, 4.19224303, 3.96448892, 4.16387305, 4.29635464,
       4.21972275, 3.90757455, 4.26161717, 4.33604402, 3.80648248,
       4.18034779, 4.31085983, 4.06092638, 4.18176833, 3.80168484,
       4.17426674, 3.77058335, 4.33077489, 4.11807656, 4.3876155 ,
       3.93193087, 4.1485797 , 4.27655134, 4.27227599, 4.03805152,
       4.09773303, 4.30444522, 3.96734781, 4.51871127, 3.92783347,
       4.25394961, 4.18133822, 4.22978961, 4.04227519, 4.15201016,
       4.10420465, 4.22520377, 4.12264176, 4.55493939, 4.17241366,
```

```
       4.06856026, 3.85934067, 4.37010508, 4.5       , 4.19177471])
```

[48]:
```
from sklearn.metrics import r2_score                    # R2 reporting for Train␣
 ↪set as value is negative model dosent seem fit
print(r2_score(y_train,y_pred))
```

```
0.15726460491857464
```

0.11 Making Predictions and Test set Report

[49]:
```
from sklearn.linear_model import LinearRegression
linear_reg=LinearRegression()
linear_reg.fit(X_test,y_test)                                   #using Linear␣
 ↪regression as a technique
print("Intercept",round(linear_reg.intercept_,3))
print("Coefficient",np.round(linear_reg.coef_,3))
y_predi=linear_reg.predict(X_test)                             #making predictions␣
 ↪using linear regression on test data set
y_predi[:100]
```

```
Intercept 4.92
Coefficient [ 0.181 -0.    -0.151 -0.007 -0.129 -0.093 -0.092 -0.187 -0.167
-0.256
 -0.302  0.563  0.531 -0.185  0.633 -0.2   -0.257  0.787 -0.138 -0.1
 -0.092 -0.364 -0.262 -0.129 -0.259  0.114 -0.126 -0.269 -0.171 -0.167
 -0.196 -0.232 -0.223 -0.174 -0.251 -0.208 -1.214 -0.952 -1.319 -0.953
 -0.    -1.273 -1.188 -0.928  0.    -0.787  0.    -0.129 -0.093 -1.119
 -0.    -0.777 -0.34  -0.092 -0.    -0.187 -1.241 -0.    -0.    -1.14
 -1.127 -1.033 -0.568 -1.032 -0.995 -0.932 -1.126 -0.167  0.    -0.256
 -0.856 -0.302 -0.815 -0.955 -0.    -0.684 -0.773 -0.69  -0.809 -1.037
 -0.    -1.759 -0.853 -0.875 -0.9   -1.073 -0.    -0.93  -0.518 -0.685
 -0.951 -1.563 -0.185 -0.2   -0.257 -0.138  0.     0.    -0.1   -0.092
 -0.121 -0.    -0.243 -0.262 -0.129 -1.352  0.    -0.458 -0.259 -0.156
  0.     0.     0.27  -0.126 -0.269 -0.171 -1.045  0.    -0.874  0.
  0.    -1.304 -1.035 -0.345 -1.162 -1.142 -0.969 -0.537 -0.167 -1.125
 -0.79   0.    -1.173 -0.196 -0.232 -1.192 -1.125 -0.638  0.     0.
 -0.223  0.    -0.174  0.    -1.161 -0.251  0.    -1.206 -0.208 -1.056
  0.     0.015 -0.017 -0.051  0.   ]
```

[49]:
```
array([4.13771425, 4.01055114, 4.27446684, 4.53740677, 4.22299271,
       4.42672691, 3.94897495, 4.24549597, 4.01299335, 4.59533933,
       3.94529146, 3.92927118, 3.76481736, 4.34374593, 4.59503736,
       4.28578238, 4.02623773, 3.98326497, 4.38036949, 3.702406  ,
       3.87315462, 4.42007194, 4.23739958, 4.15785105, 4.54721422,
       4.54548197, 3.97731777, 3.92384513, 4.4982522 , 4.36498421,
       3.80372029, 4.1978003 , 4.38814783, 4.59174014, 4.23273617,
```

```
       4.0058805 , 3.99205984, 3.86607299, 3.97075367, 3.89118635,
       4.23165566, 3.90080827, 4.00514756, 4.07291936, 3.80235705,
       4.66227993, 3.88897821, 4.14192584, 3.89202855, 4.32550951,
       3.98208092, 3.9986708 , 4.30301436, 3.88608694, 4.35109658,
       4.03006183, 3.87640468, 4.50041266, 4.18474848, 3.90489045,
       3.72453375, 4.56380388, 4.27061413, 4.04181427, 4.38414341,
       4.33019728, 4.47243549, 4.21549371, 3.93577763, 4.5175637 ,
       3.94177591, 4.1285773 , 3.75437462, 4.14468801, 4.01256275,
       4.25783919, 3.88610432, 4.24679113, 3.1       , 3.65461376,
       4.42152845, 4.03494134, 3.9898242 , 4.05249002, 4.43682494,
       4.2357062 , 3.90053457, 4.10188474, 3.99345226, 4.55949963,
       3.78303249, 4.24172205, 4.13404017, 4.53618183, 4.30838564,
       4.47587171, 4.31227573, 4.40170164, 4.43865955, 4.04788512])
```

[50]:
```
from sklearn.metrics import r2_score                    #R2 is negative we can␣
 ↪say data is varied to predict from this model
print (r2_score(y_test, y_predi))
```

```
0.18974390832275345
```

1 Linear Regression Model is not suitable for Predicting this Data

1.1 Thank You

[ ]:
```

```