****

# Python Data Science project

# Machine Learning

# Topic - App Rating Prediction

**BY :PRIYA MISHRA**

**Problem Statement: DESCRIPTION Objective: Make a model to predict the app rating, with other information about the app provided.**

**# Importing library**

import **numpy** as **np**

import **pandas** as **pd**

import **seaborn** as **sns**

import **matplotlib.pyplot** as **plt**

%matplotlib **inline**

**#1. Load the data file using pandas**

**df** = **pd**.**read\_csv('googleplaystore.csv')**

output: **#Displays the first five data from the data file**

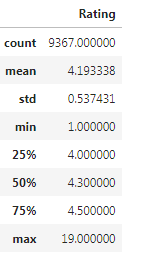
df.head()



#to get statistical info

df.describe()

**output :**

****

#Shape gives the total number of rows and the columns present in the dataframe

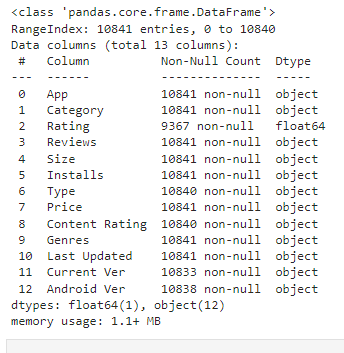
df.shape

**output :** **#Observations:** The dataframe contains 10841 rows and 13 columns

**#prints the information about the dataframe**

df.info()

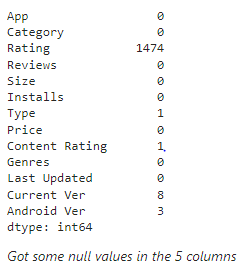
**output :**

****

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

df.isnull().sum()

**output :**

****

**#observation : Displays the total number of null values present in the each column**

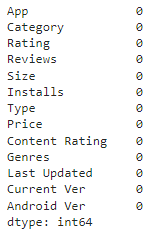
**Data Wrangling**

**#1 Dropping all the Null values present in the dataframe**

df.dropna(inplace=True)

# Checking the revised Rows and columns

df.isnull().sum

**.**

**#Observations: None of the columns have null values**

**# Checking the revised Rows and columns**

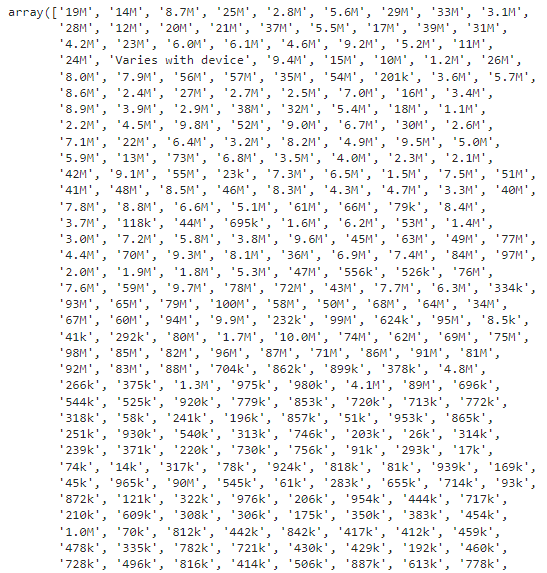
df.reset\_index(drop= True, inplace = True)

df.shape

**output :**

****

df['Size'].unique()

**output :** 

**As the model do not understand categorical variable so before moving towards the visualization all categorical Data types must be converted to numeric on which the analysis is to be done**

**#I) Start the cleaning with Size Column and converting in to numeric**

df['Size'] = df['Size'].apply(lambda x: str(x).replace('M', '') if 'M' in str(x) else x)

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

**# Scaling all the values to Millions format (means that 19.0 => 19x10^6 => 19M)**

df['Size'] = df['Size'].apply(lambda x: float(str(x).replace('k', ''))/1000 if 'k' in str(x) else x)

df['Size'] = df['Size'].apply(lambda x : float(x))

df = df[pd.notnull(df['Size'])]

df['Size'].dtype

output :



**#observation : df['Size'] type changed from object to float**

df.shape

output :



df['Reviews'].unique()

**output :**

****

**#II) Converting the Reviews column**

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

df['Reviews'].dtype

**output :**

****

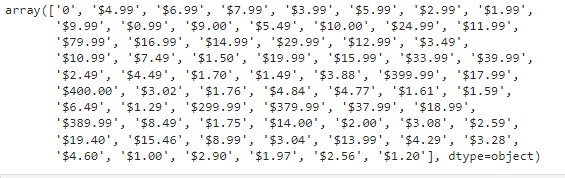
df['Rating'].dtype

**output :**

****

df['Price'].unique()

**output :**

****

**#III) Moving on with the Cleaning and conversion of Price column**

df['Price'] = df['Price'].apply((lambda x:str(x).replace('$','') if '$' in str(x) else str(x)))

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

df['Price'].dtype

**output :**

****

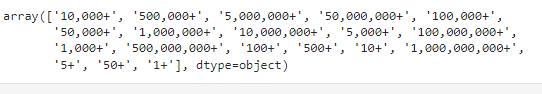
df.shape

**output :**

****

df['Installs'].unique()

**output :**

****

**#IV) Cleaning and conversion of the Installs column**

df['Installs'] = df['Installs'].apply (lambda x: str(x).replace('+','') if '+' in str(x) else x)

df['Installs'] = df['Installs'].apply(lambda x: str(x).replace(',', '') if ',' in str(x) else x)

df['Installs'] = df['Installs'].apply(lambda x: int (x))

df['Installs'].dtype

**output :**

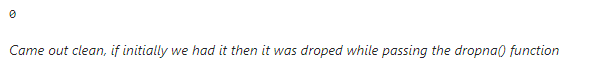
dtype('int64')

df.shape

**output :**

****

df[df['Rating']>5].shape[0]

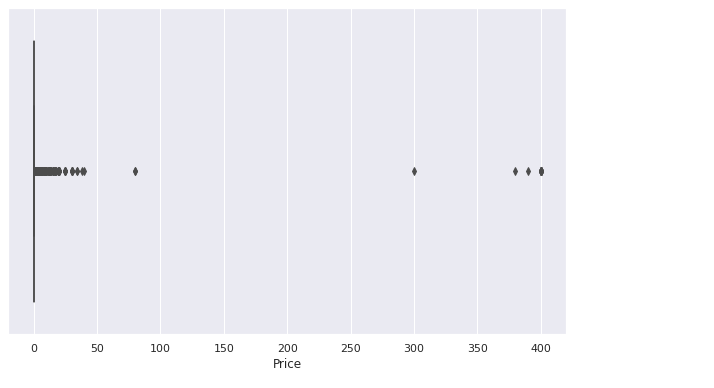
**output : **

#5.Univariate Analysis

# Box Plot for Price

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

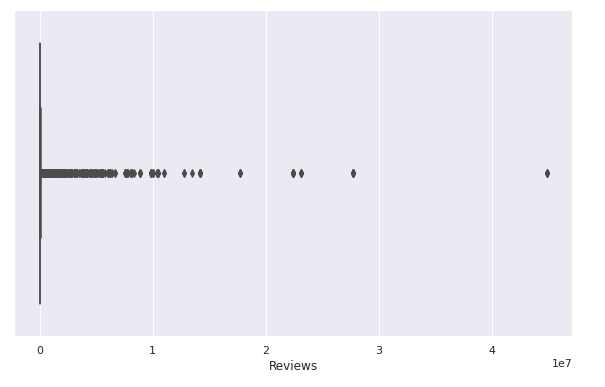
sns.boxplot(x= 'Price',data= df);

****

**#observation : There are some outliers in the Price column,i.e., there are some apps whose price is more than usual apps on the Googleplaystore**

# Boxplot for Reviews

sns.boxplot(x ='Reviews', data =df);

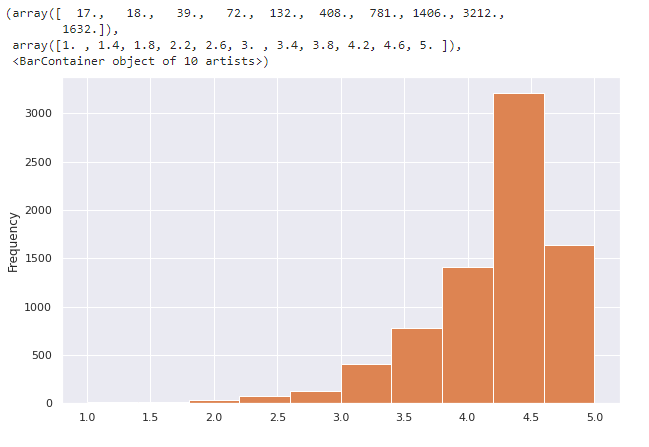
****

**observation : There are some apps that have very high number of Reviews.**

# Histogram for Rating

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

plt.hist(df['Rating'])

****

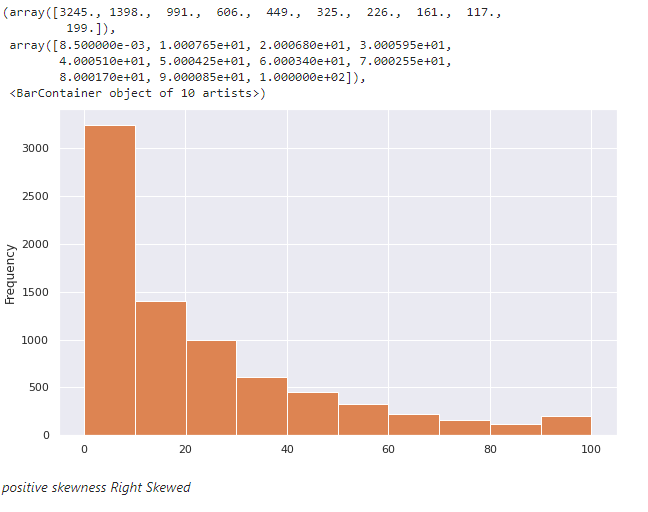
**#observation: There is a Negative skewness(left- skewed)**

**#some apps seem to have higher Ratings than usual**

# Histogram for Size

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

plt.hist(df['Size'])

****

**#observation : positive skewness Right Skewed**

#6.Handling Outliers

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

**#In the Installs column as well**

**#I) price of $200 and above for an application is expected to be very high**

**#we can use either to get the results**

df[df['Price']>200].index.shape[0]

df.loc[df['Price']>200].shape[0]

**output :**

****

#Dropping the Junk apps

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

3

**# rechecking No of rows and columns after drop**

df.shape

output :

****

#II) Very few apps have very high no. of Reviews

df.loc[df['Reviews']>2000000].shape[0]

**output :**

****

#Dropping the Star apps as these will skew the analysis,

#checking the shape after dropping

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

df.shape

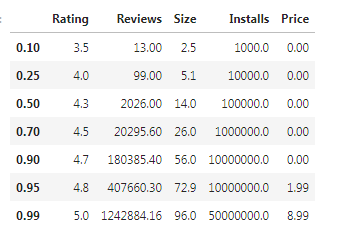
**ouput :**

****

Find out the Percentiles of Installs and decide a threshold as cutoff for outlier

# df.quantile([0.1,0.25,0.5,0.7,0.9,0.95,0.99], axis=0)

**output :**

****

#dropping the value more than the **cutoff(**threshold **-95th percentile)**

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

# df.shape

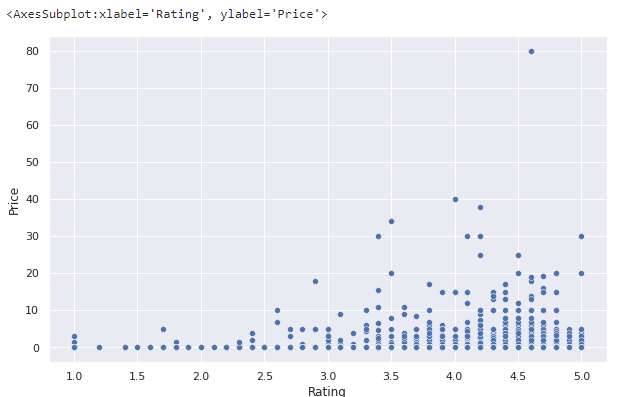
**output :**

****

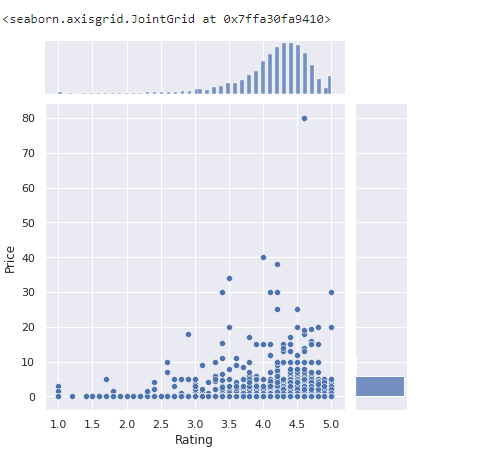
#7.Bivariate analysis

**#1) Scatter plot/jointplot for Rating Vs. Price**

# sns.scatterplot(x = 'Rating', y = 'Price',data=df)

****

# sns.jointplot(x= 'Rating',y= 'Price',data= df)

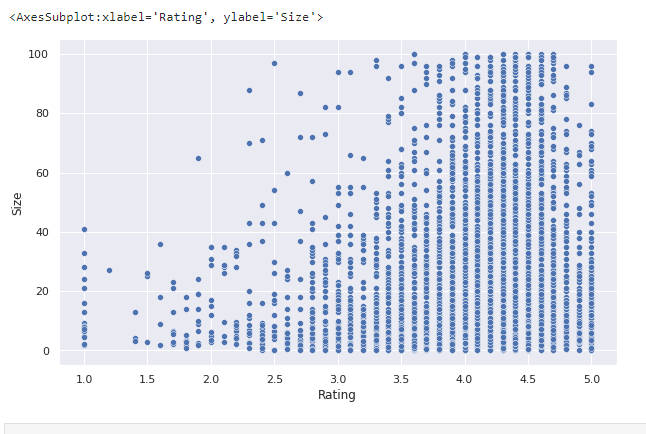
****

**Observation :**

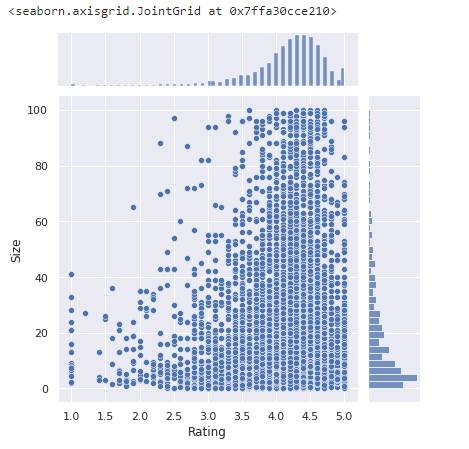
Both the plots show a positive linear relationship; as the price of an app increases its rating also increases. That states the paid apps have the highest of Ratings

#2) Scatterplot/jointplot for Rating Vs. Size

# sns.scatterplot(x= 'Rating',y= 'Size', data= df)

****

# sns.jointplot(x= 'Rating', y= 'Size', data= df)

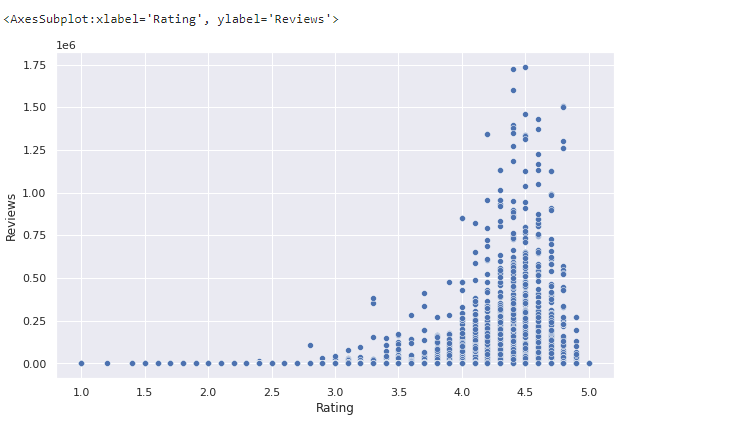


Observation :

**The plots show a positive linear relationship; as the Size increases the Ratings increases. This stats the heavier apps are rated better**

**Scatterplot for Ratings Vs. Reviews**

# sns.scatterplot(x= 'Rating',y= 'Reviews', data= df)



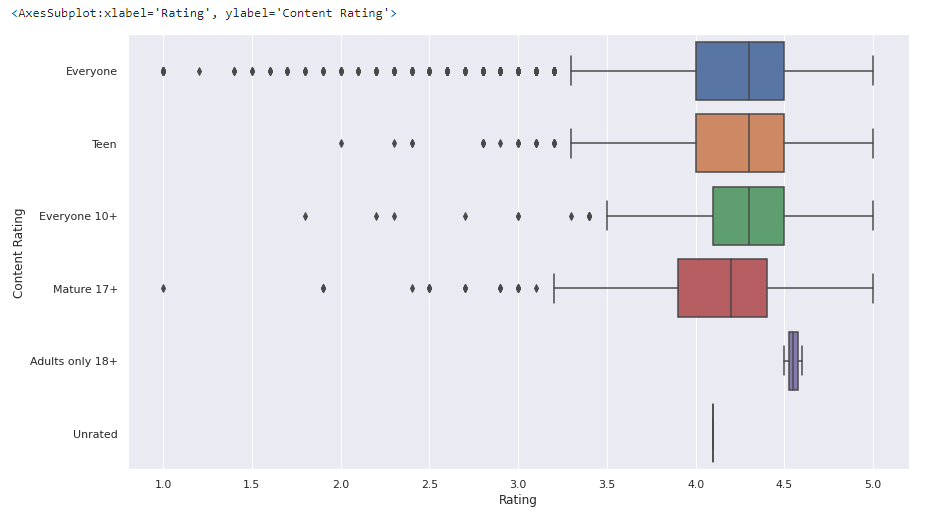
**Observation :**

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

Boxplot for Ratings Vs. Content Rating

# sns.set(rc={'figure.figsize':(14,8)})

# sns.boxplot(x= 'Rating', y= 'Content Rating',data = df)

****

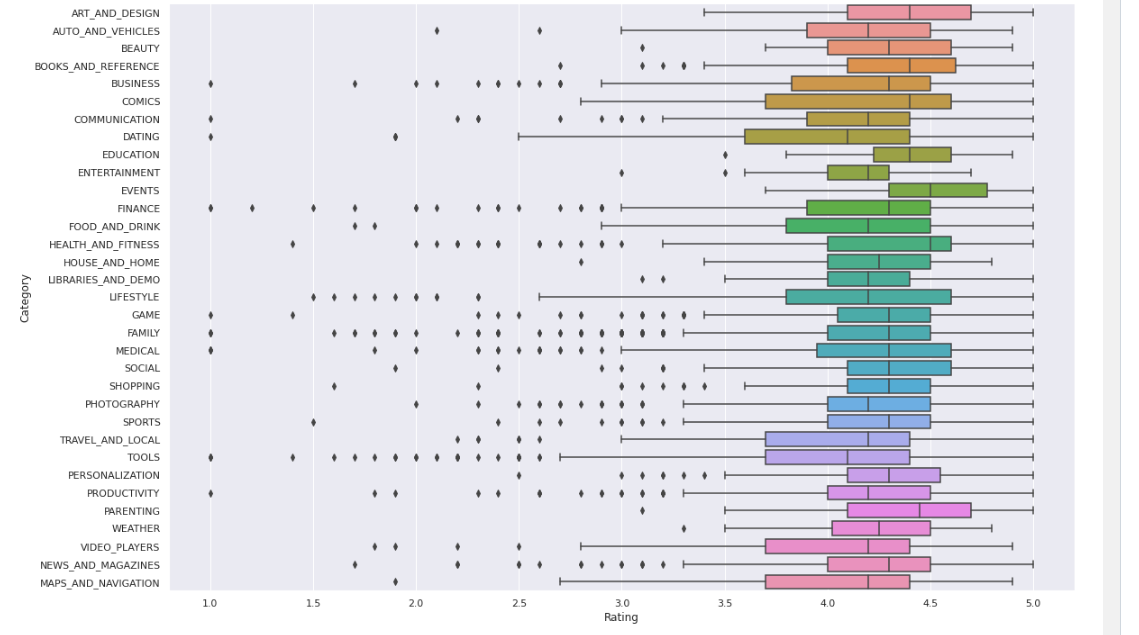
Observation:

**The above plot shows the apps for Everyone is worst rated as it contain the highest number of outliers followed by apps for Mature 17+ and Everyone 10+ along with Teen. The category Adults only 18+ is rated better and falls under most liked type**

**Boxplot for Ratings Vs. Category**

# sns.set(rc={'figure.figsize':(18,12)})

# sns.boxplot(x= 'Rating', y = 'Category', data= df)

****

**Observation :**

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

# 8.Data Preprocessing

# ****Model development****

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

# inp1= df.copy()

# inp1.head()

# output :

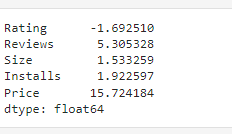
# 

# Observation :

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

# inp1.skew()

**output :**

****

**#1) apply log transformation to Reviews**

# reviews\_skew = np.log1p(inp1['Reviews'])

# inp1['Reviews']= reviews\_skew

# reviews\_skew.skew()

output :

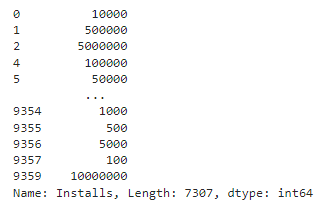
****

**#2) apply log transformation to Installs**

Installs\_skew = np.log1p(inp1['Installs'])

inp1['Installs']

**output :**



Installs\_skew.skew()

**Output :**



1.inp1.head()

**output :**

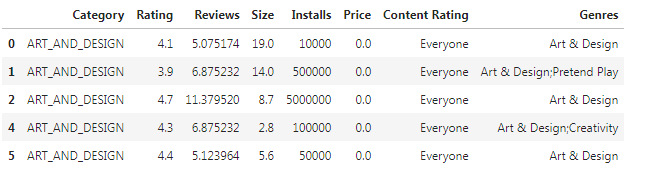


**#2) Dropping the columns- App, Last Updated, Current Ver, Type, & Andriod Ver as these won't be useful for our model**

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

inp1.head()

**output :**

****

inp1.shape

**output :**

****

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

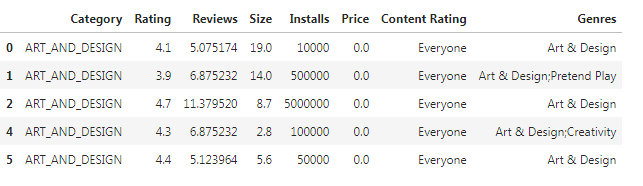
**Dummy Encoding is one way to convert these columns into numerical**

**#3) create a copy of dataframe**

inp2 = inp1

inp2.head()

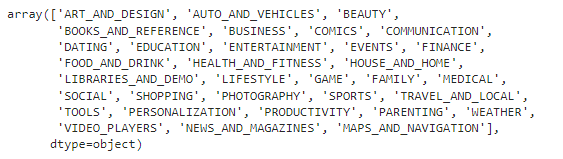
**output :**

****

**#get unique values in column category**

inp2['Category'].unique()

**output :**

****

inp2.Category = pd.Categorical(inp2.Category)

x = inp2[['Category']]

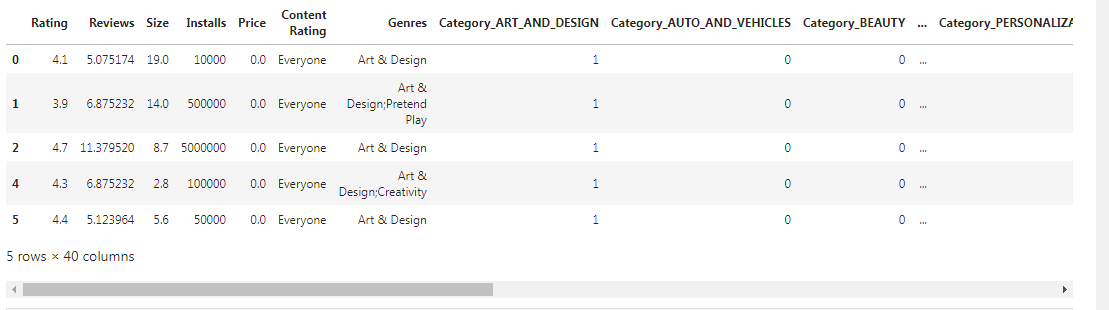
del inp2['Category']

dummies = pd.get\_dummies(x, prefix = 'Category')

inp2 = pd.concat([inp2,dummies], axis=1)

inp2.head()

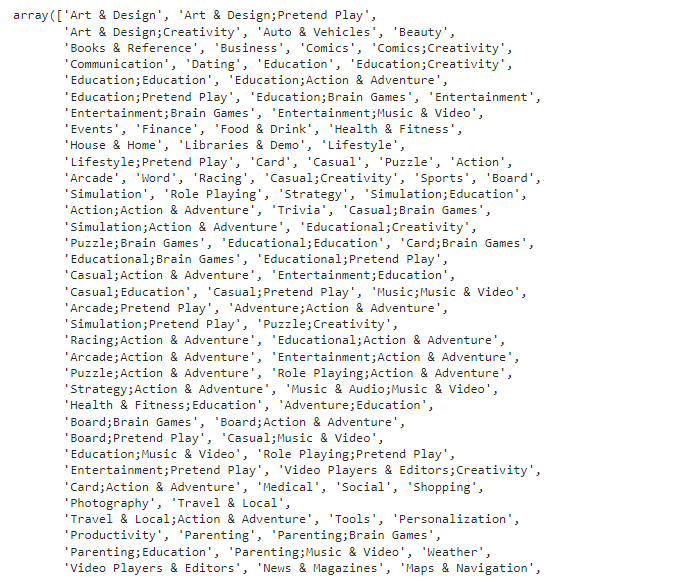
**output :**

****

**#get unique values in Column Genres**

inp2["Genres"].unique()

**output :**

****

**Observation :**

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

**#Create an empty list**

lists = []

#Get the total genres count and gernes count of perticular gerner count less than 20 append those into the list

for i in inp2.Genres.value\_counts().index:

if inp2.Genres.value\_counts()[i]<20:

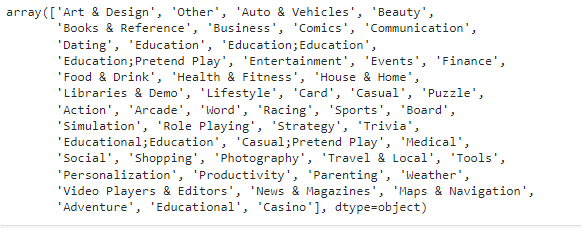
lists.append(i)

#changing the gerners which are in the list to other

inp2.Genres = ['Other' if i in lists else i for i in inp2.Genres]

inp2["Genres"].unique()

**output :**

****

**#Storing the genres column into x varible and delete the genres col from dataframe inp2**

**#And concat the encoded cols to the dataframe inp2**

inp2.Genres = pd.Categorical(inp2['Genres'])

x = inp2[["Genres"]]

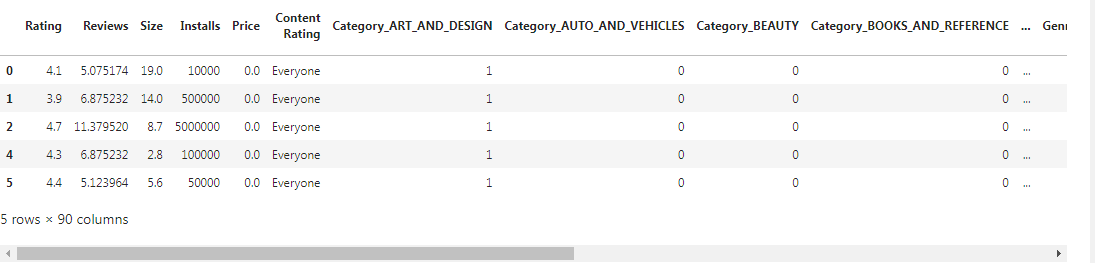
del inp2['Genres']

dummies = pd.get\_dummies(x, prefix = 'Genres')

inp2 = pd.concat([inp2,dummies], axis=1)

inp2.head()

**output :**



#**getting the unique values in Column "Content Rating"**

inp2["Content Rating"].unique()

**output :**



**#Applying one hot encoding**

**#Storing the Content Rating column into x varible and delete the Content Rating col from dataframe inp2**

**#And concat the encoded cols to the dataframe inp2**

inp2['Content Rating'] **=** pd**.**Categorical(inp2['Content Rating'])

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

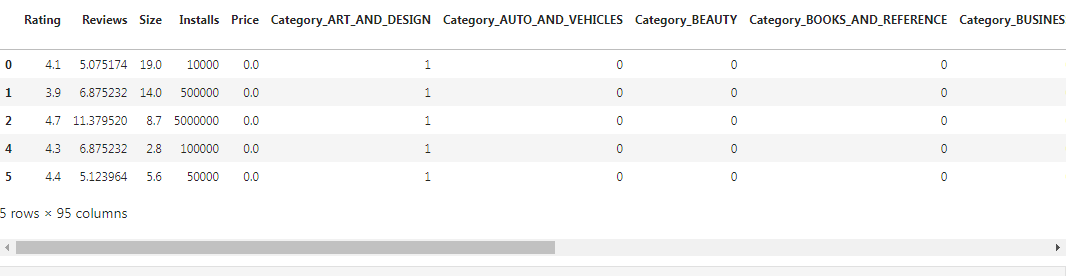
**del** inp2['Content Rating']

dummies **=** pd**.**get\_dummies(x, prefix **=** 'Content Rating')

inp2 **=** pd**.**concat([inp2,dummies], axis**=**1)

inp2**.**head()

**output :**



inp2.shape

**output :**

****

**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**

*#importing the neccessary libraries from sklearn to split the data and and for model building*

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.metrics **import** mean\_squared\_error **as** mse

**from** sklearn **import** metrics

**#Creating the variable X and Y which contains the X features as independent features and Y is the target feature**

df2 = inp2

X **=** df2**.**drop('Rating',axis**=**1)

y **=** df2['Rating']

**#Dividing the X and y into test and train data**

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y, test\_size**=**0.3, random\_state**=**5)

# Model Building & Evaluation

**11. Model building Use linear regression as the technique Report the R2 on the train set**

**#Create a linear reggression obj by calling the linear reggressor algorithm**

lin\_reggressor **=** LinearRegression()

lin\_reggressor**.**fit(X\_train,y\_train)

LinearRegression()

R2\_Score\_train\_data **=** round(lin\_reggressor**.**score(X\_train,y\_train),3)

print("The R2 value of the Training Set is : {}"**.**format(R2\_Score\_train\_data))

The R2 value of the Training Set is : 0.068

Make predictions on test set and report R2.

*# test the output by changing values, like 3750*

y\_pred **=** lin\_reggressor**.**predict(X\_test)

R2\_Score\_test\_data **=**metrics**.**r2\_score(y\_test,y\_pred)

R2\_Score\_test\_data

**output :**

****

R2\_Score\_test\_data **=** round(lin\_reggressor**.**score(X\_test,y\_test),3)

print("The R2 value of the Training Set is : {}"**.**format(R2\_Score\_test\_data))

The R2 value of the Training Set is : 0.058