In [133…
```python
#importing libaraires
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('vgsales.csv')
```

Understanding the Data

In [134…
```python
#exploring the data

df.describe()
```

Out[134]:

	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
count	16598.000000	16327.000000	16598.000000	16598.000000	16598.000000	16598.000000	16598.000000
mean	8300.605254	2006.406443	0.264667	0.146652	0.077782	0.048063	0.537441
std	4791.853933	5.828981	0.816683	0.505351	0.309291	0.188588	1.555028
min	1.000000	1980.000000	0.000000	0.000000	0.000000	0.000000	0.010000
25%	4151.250000	2003.000000	0.000000	0.000000	0.000000	0.000000	0.060000
50%	8300.500000	2007.000000	0.080000	0.020000	0.000000	0.010000	0.170000
75%	12449.750000	2010.000000	0.240000	0.110000	0.040000	0.040000	0.470000
max	16600.000000	2020.000000	41.490000	29.020000	10.220000	10.570000	82.740000

In [135…
```python
df.shape
```

Out[135]:
```
(16598, 11)
```

In [136…
```python
df.columns
```

Out[136]:
```
Index(['Rank', 'Name', 'Platform', 'Year', 'Genre', 'Publisher', 'NA_Sales',
       'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales'],
      dtype='object')
```

In [137…
```python
df.dtypes
```

Out[137]:
```
Rank              int64
Name             object
Platform         object
Year            float64
Genre            object
Publisher        object
NA_Sales        float64
EU_Sales        float64
JP_Sales        float64
Other_Sales     float64
Global_Sales    float64
dtype: object
```

In [138…
```python
df.head()
```

Out[138]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37

In [139…
```python
df.tail()
```

Out[139]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
16593	16596	Woody Woodpecker in Crazy Castle 5	GBA	2002.0	Platform	Kemco	0.01	0.00	0.0	0.0	0.01
16594	16597	Men in Black II: Alien Escape	GC	2003.0	Shooter	Infogrames	0.01	0.00	0.0	0.0	0.01
16595	16598	SCORE International Baja 1000: The Official Game	PS2	2008.0	Racing	Activision	0.00	0.00	0.0	0.0	0.01
16596	16599	Know How 2	DS	2010.0	Puzzle	7G//AMES	0.00	0.01	0.0	0.0	0.01
16597	16600	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	0.00	0.0	0.0	0.01

In [140…
```python
df.info()
```

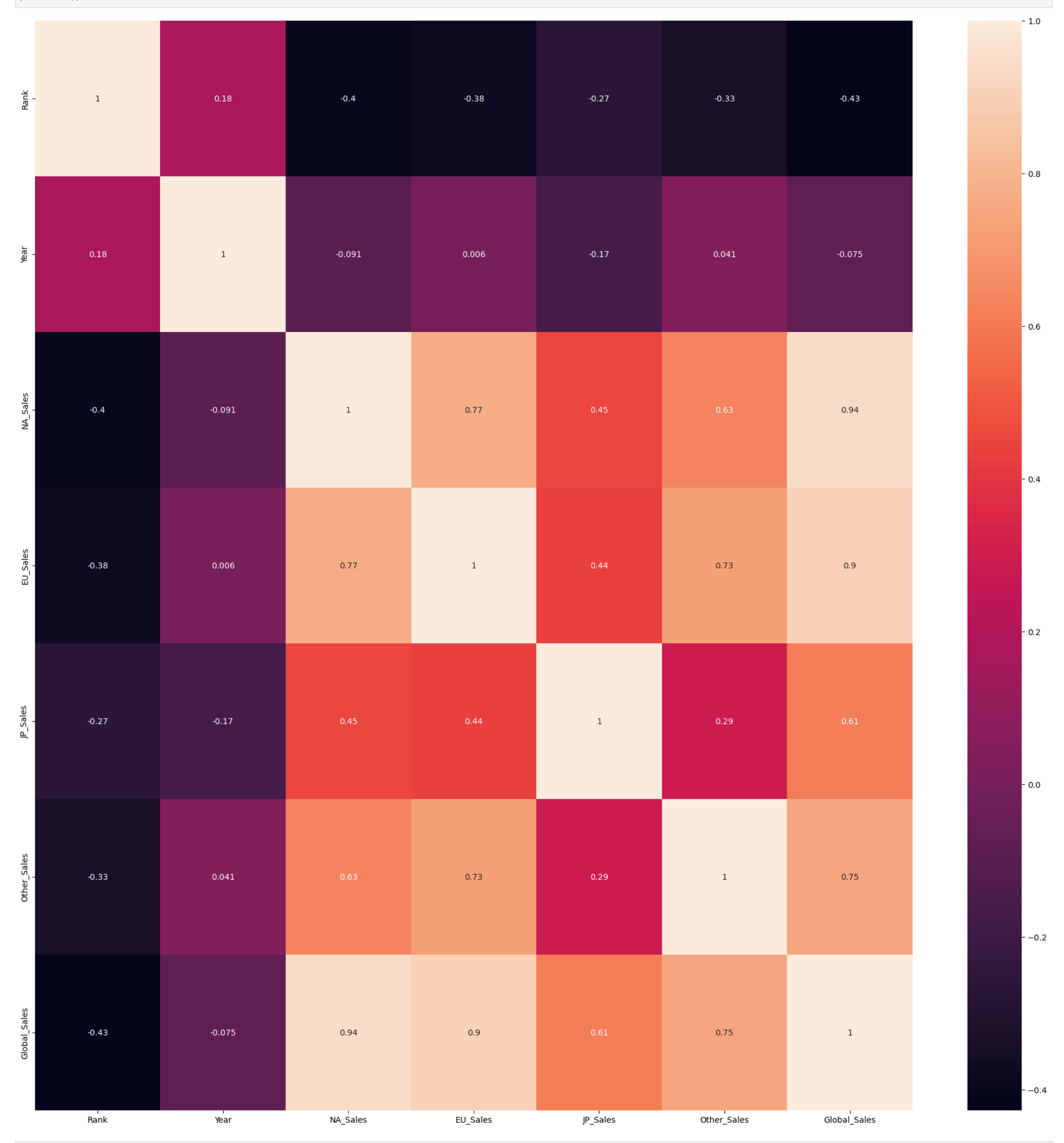
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16598 entries, 0 to 16597
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Rank          16598 non-null  int64  
 1   Name          16598 non-null  object 
 2   Platform      16598 non-null  object 
 3   Year          16327 non-null  float64
 4   Genre         16598 non-null  object 
 5   Publisher     16540 non-null  object 
 6   NA_Sales      16598 non-null  float64
 7   EU_Sales      16598 non-null  float64
 8   JP_Sales      16598 non-null  float64
 9   Other_Sales   16598 non-null  float64
 10  Global_Sales  16598 non-null  float64
dtypes: float64(6), int64(1), object(4)
memory usage: 1.4+ MB
```

In [141…
```python
df.isnull().sum()
```

Out[141]:
```
Rank              0
Name              0
Platform          0
Year            271
Genre             0
Publisher        58
NA_Sales          0
EU_Sales          0
JP_Sales          0
Other_Sales       0
Global_Sales      0
dtype: int64
```

bivirate analysis

In [142…
```python
# applying bivirate analysis to find the correlation using heat map
plt.figure(figsize=(24,24))
df_corr = df.corr()
sns.heatmap(df_corr,annot=True)
plt.show()
```



In [143…
```python
cat,num = [],[]

for columns in df.columns:
    if df[columns].dtype=='object':
        cat.append(columns)
    else:
        num.append(columns)
print('Category:',cat )
print('Numeric:',num)
```

```
Category: ['Name', 'Platform', 'Genre', 'Publisher']
Numeric: ['Rank', 'Year', 'NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales']
```

In [144…
```python
def correlation(feature):
    dictionary = dict(df.corrwith(df[feature]))

    corrdict = {}

    for key, value in dictionary.items():
        if (value > 0.80 or value < -0.97) and (feature != key):
            corrdict[feature] = key
            print(feature,"------",key)

print("There is a strong correlation with")
for column in num:
    correlation(column)
```

```
There is a strong correlation with
NA_Sales ------ Global_Sales
EU_Sales ------ Global_Sales
Global_Sales ------ NA_Sales
Global_Sales ------ EU_Sales
```

In [145…
```python
df.isnull().sum()
```

Out[145]:
```
Rank              0
Name              0
Platform          0
Year            271
Genre             0
Publisher        58
NA_Sales          0
EU_Sales          0
JP_Sales          0
Other_Sales       0
Global_Sales      0
dtype: int64
```

In [146…
```python
df.shape
```

Out[146]:
```
(16598, 11)
```

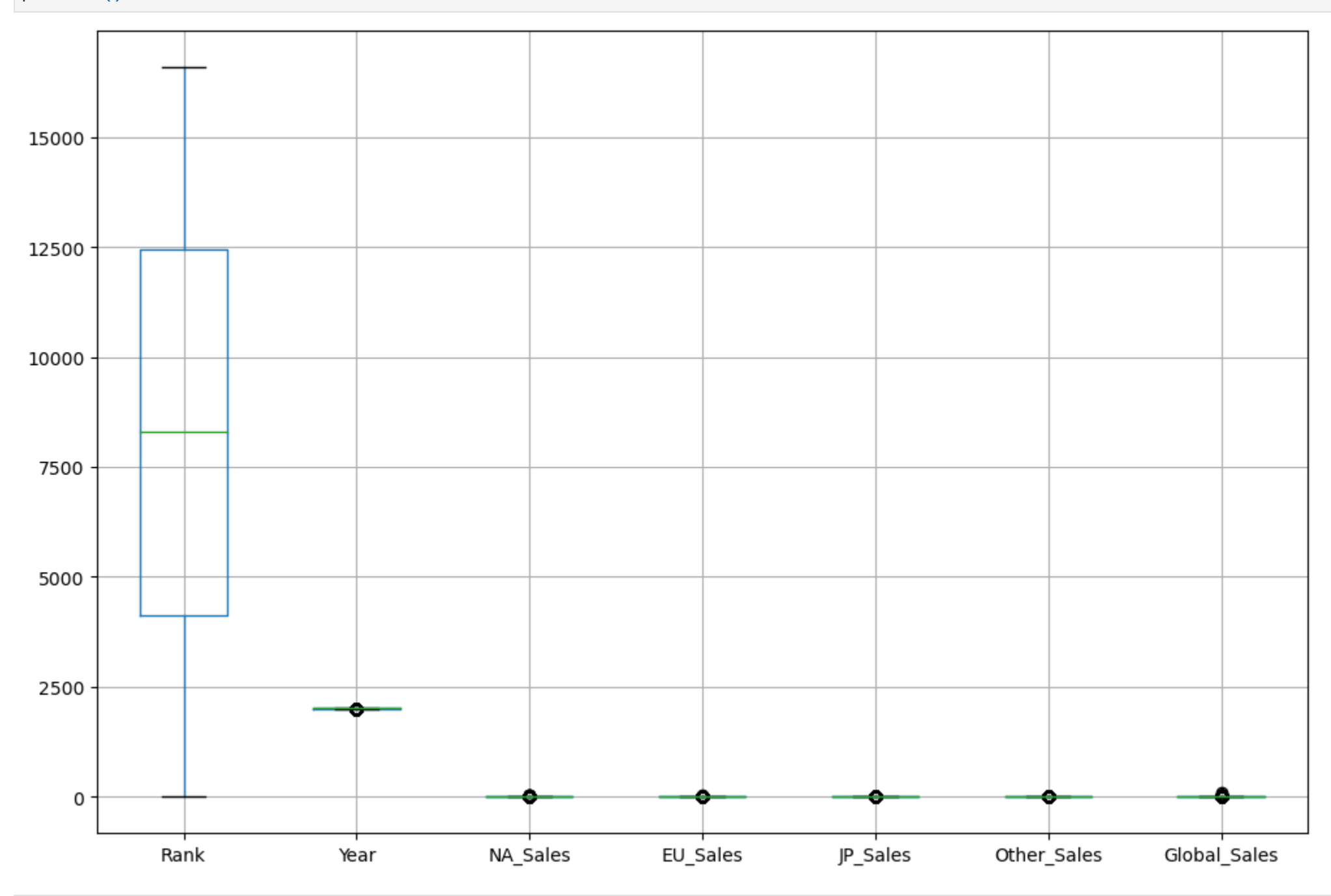
In [147…
```python
df = df.dropna(subset=["Year"])
df = df.dropna(subset=["Publisher"])
```

In [148…
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16291 entries, 0 to 16597
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Rank          16291 non-null  int64  
 1   Name          16291 non-null  object 
 2   Platform      16291 non-null  object 
 3   Year          16291 non-null  float64
 4   Genre         16291 non-null  object 
 5   Publisher     16291 non-null  object 
 6   NA_Sales      16291 non-null  float64
 7   EU_Sales      16291 non-null  float64
 8   JP_Sales      16291 non-null  float64
 9   Other_Sales   16291 non-null  float64
 10  Global_Sales  16291 non-null  float64
dtypes: float64(6), int64(1), object(4)
memory usage: 1.5+ MB
```

Outliers

In [149…
```python
plt.figure(figsize=(12,8))
df.boxplot()
plt.show()
```



In [150…
```python
# setting the limit for the outlier
outlier_list = []
def outlier_detection(data):
    threshold = 4
    mean_avg = np.mean(data)
    stdeviation = np.std(data)

    for variable in data:
        z_score = (variable - mean_avg)/stdeviation
        if np.abs(z_score) > threshold:
            outlier_list.append(variable)
    return outlier_list
```

In [151…
```python
num = []
for element in df.columns:
    if df[element].dtype != "object":
        num.append(element)
print(f"numerical data -> {num}")
```

```
numerical data -> ['Rank', 'Year', 'NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales']
```

In [152…
```python
for variable in num:
    outliers = outlier_detection(df[variable])
    if len(outliers) > 0:
        print(f"Number of Outliers for {variable} --> {len(outliers)}")
```

```
Number of Outliers for Year --> 108
Number of Outliers for NA_Sales --> 214
Number of Outliers for EU_Sales --> 340
Number of Outliers for JP_Sales --> 493
Number of Outliers for Other_Sales --> 601
Number of Outliers for Global_Sales --> 718
```

In [153…
```python
IQR = df["Rank"].quantile(0.75) - df["Rank"].quantile(0.25)
lower_limit = df["Rank"].quantile(0.25)-(IQR*1.5)
upper_limit = df["Rank"].quantile(0.75)+(IQR*1.5)
print(f"Upper Limit is {upper_limit}")
print(f"Lower Limit is {lower_limit}")
```

```
Upper Limit is 24900.0
Lower Limit is -8328.0
```

In [154…
```python
df.loc[df["Rank"]<lower_limit,"Rank"] = lower_limit
df.loc[df["Rank"]>upper_limit,"Rank"] = upper_limit
```

In [155…
```python
IQR = df["Year"].quantile(0.75) - df["Year"].quantile(0.25)
lower_limit = df["Year"].quantile(0.25)-(IQR*1.5)
upper_limit = df["Year"].quantile(0.75)+(IQR*1.5)
print(f"Upper Limit is {upper_limit}")
print(f"Lower Limit is {lower_limit}")
df.loc[df["Year"]<lower_limit,"Year"] = lower_limit
df.loc[df["Year"]>upper_limit,"Year"] = upper_limit
```

```
Upper Limit is 2020.5
Lower Limit is 1992.5
```

In [156…
```python
IQR = df["NA_Sales"].quantile(0.75) - df["NA_Sales"].quantile(0.25)
lower_limit = df["NA_Sales"].quantile(0.25)-(IQR*1.5)
upper_limit = df["NA_Sales"].quantile(0.75)+(IQR*1.5)
print(f"Upper Limit is {upper_limit}")
print(f"Lower Limit is {lower_limit}")
df.loc[df["NA_Sales"]<lower_limit,"NA_Sales"] = lower_limit
df.loc[df["NA_Sales"]>upper_limit,"Year"] = upper_limit
```

```
Upper Limit is 0.6
Lower Limit is -0.36
```

In [157…
```python
IQR = df["EU_Sales"].quantile(0.75) - df["EU_Sales"].quantile(0.25)
lower_limit = df["EU_Sales"].quantile(0.25)-(IQR*1.5)
upper_limit = df["EU_Sales"].quantile(0.75)+(IQR*1.5)
print(f"Upper Limit is {upper_limit}")
print(f"Lower Limit is {lower_limit}")
df.loc[df["EU_Sales"]<lower_limit,"EU_Sales"] = lower_limit
df.loc[df["EU_Sales"]>upper_limit,"EU_Sales"] = upper_limit
```

```
Upper Limit is 0.275
Lower Limit is -0.165
```

In [158…
```python
IQR = df["JP_Sales"].quantile(0.75) - df["JP_Sales"].quantile(0.25)
lower_limit = df["JP_Sales"].quantile(0.25)-(IQR*1.5)
upper_limit = df["JP_Sales"].quantile(0.75)+(IQR*1.5)
print(f"Upper Limit is {upper_limit}")
print(f"Lower Limit is {lower_limit}")
df.loc[df["JP_Sales"]<lower_limit,"JP_Sales"] = lower_limit
df.loc[df["JP_Sales"]>upper_limit,"JP_Sales"] = upper_limit
```

```
Upper Limit is 0.1
Lower Limit is -0.06
```

In [159…
```python
IQR = df["Other_Sales"].quantile(0.75) - df["Other_Sales"].quantile(0.25)
lower_limit = df["Other_Sales"].quantile(0.25)-(IQR*1.5)
upper_limit = df["Other_Sales"].quantile(0.75)+(IQR*1.5)
print(f"Upper Limit is {upper_limit}")
print(f"Lower Limit is {lower_limit}")
df.loc[df["Other_Sales"]<lower_limit,"Other_Sales"] = lower_limit
df.loc[df["Other_Sales"]>upper_limit,"Other_Sales"] = upper_limit
```

```
Upper Limit is 0.1
Lower Limit is -0.06
```

In [160…
```python
IQR = df["Global_Sales"].quantile(0.75) - df["Global_Sales"].quantile(0.25)
lower_limit = df["Global_Sales"].quantile(0.25)-(IQR*1.5)
upper_limit = df["Global_Sales"].quantile(0.75)+(IQR*1.5)
print(f"Upper Limit is {upper_limit}")
print(f"Lower Limit is {lower_limit}")
df.loc[df["Global_Sales"]<lower_limit,"Global_Sales"] = lower_limit
df.loc[df["Global_Sales"]>upper_limit,"Global_Sales"] = upper_limit
```

```
Upper Limit is 1.1099999999999999
Lower Limit is -0.5700000000000001
```

Converting Categorical data into Integer values

In [161…
```python
#converting strings into integers
categorical, numerical = [],[]

for features in df:
    if df[features].dtype == "object":
        categorical.append(features)
    else:
        numerical.append(features)

print(f"numerical data {numerical}")
print(f"categorical data {categorical}")
```

```
numerical data ['Rank', 'Year', 'NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales']
categorical data ['Name', 'Platform', 'Genre', 'Publisher']
```

In [162…
```python
df = df.drop("Name",axis=1)
df = df.drop("Publisher",axis=1)
```

In [163…
```python
platform_int = pd.get_dummies(df["Platform"],drop_first=True)
platform_int.head()
```

Out[163]:

	3DO	3DS	DC	DS	GB	GBA	GC	GEN	GG	N64	...	SAT	SCD	SNES	TG16	WS	Wii	WiiU	X360	XB	XOne
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 30 columns

In [164…
```python
genre_int = pd.get_dummies(df["Genre"],drop_first=True)
genre_int.head()
```

Out[164]:

	Adventure	Fighting	Misc	Platform	Puzzle	Racing	Role-Playing	Shooter	Simulation	Sports	Strategy
0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	1	0	0	0	0

In [165…
```python
new_df = pd.concat([df,platform_int,genre_int,publisher_int],axis=1)
new_df = new_df.drop(["Platform","Genre"],axis=1)
new_df.head()
```

Out[165]:

	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	3DO	3DS	DC	...	Zushi Games	bitComposer Games	dramatic create	fonfun	iWin	id Software	imageepoch Inc.	inXile Entertainment	mixi, Inc	responDESIGN
0	1	0.6	41.49	0.275	0.1	0.1	1.11	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	2	0.6	29.08	0.275	0.1	0.1	1.11	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	3	0.6	15.85	0.275	0.1	0.1	1.11	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	4	0.6	15.75	0.275	0.1	0.1	1.11	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	5	0.6	11.27	0.275	0.1	0.1	1.11	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 622 columns

Machine Learning Model Training

In [166…
```python
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(new_df.drop("Global_Sales",axis=1),new_df["Global_Sales"],test_size=0.2,random_state=1)
```

In [167…
```python
#importing libraries for machine learning
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.metrics import r2_score
import numpy as np
from sklearn.metrics import accuracy_score
```

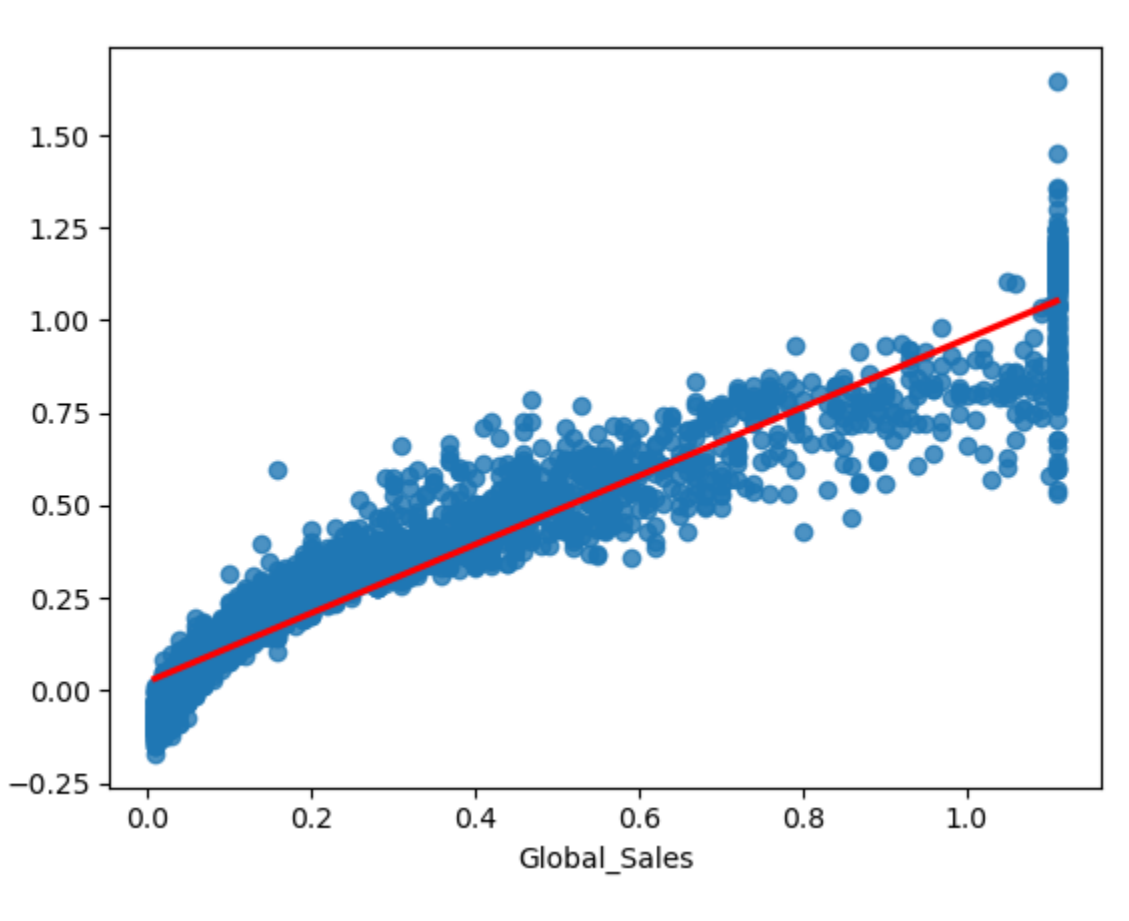
Linear Regression

In [168…
```python
#performing linear regression
from sklearn.linear_model import LinearRegression
linear_reg = LinearRegression()
linear_reg.fit(X_train,y_train)
```

Out[168]:
```
LinearRegression()
```

In [169…
```python
linear_ypred = linear_reg.predict(X_test)
sns.regplot(x=y_test,y=linear_ypred,line_kws={"color":"red"})
```

Out[169]:
```
<AxesSubplot:xlabel='Global_Sales'>
```



In [170…
```python
mae = mean_absolute_error(y_test,linear_ypred)
mse = mean_squared_error(y_test,linear_ypred)
r2 = r2_score(y_test,linear_ypred)

print(f"Mean Absolute Error is {mae}")
print(f"Mean Squared Error is {mse}")
print(f"R Square is {r2}")
```

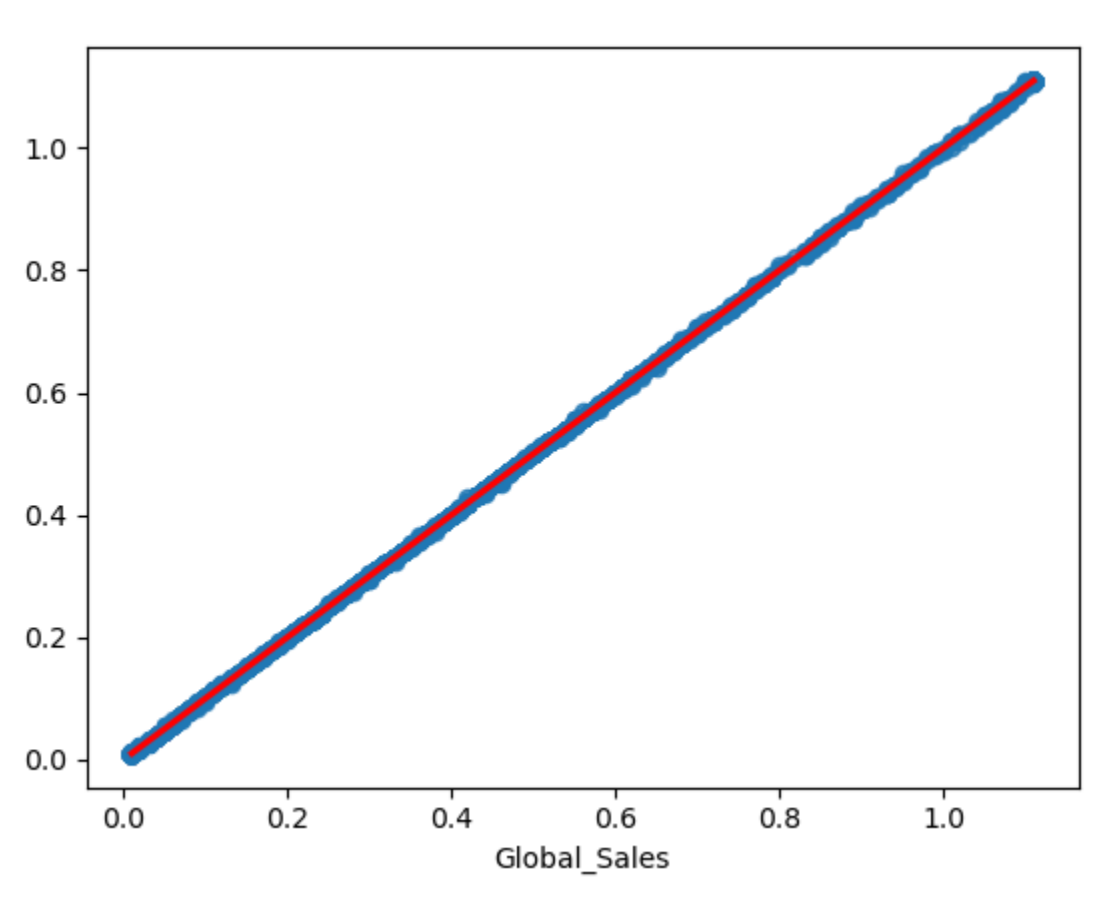
```
Mean Absolute Error is 0.07110871434579838
Mean Squared Error is 0.009843503531106487
R Square is 0.9254923560248172
```

K Neighbors Classifier

In [171…
```python
#applying KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsRegressor(n_neighbors=4)
knn.fit(X_train,y_train)
knn_y = knn.predict(X_test)
```

In [172…
```python
sns.regplot(x=y_test,y=knn_y,line_kws={"color":"red"})
```

Out[172]:
```
<AxesSubplot:xlabel='Global_Sales'>
```



In [173…
```python
mae = mean_absolute_error(y_test,knn_y)
mse = mean_squared_error(y_test,knn_y)
r2 = r2_score(y_test,knn_y)

print(f"Mean Absolute Error is {mae}")
print(f"Mean Squared Error is {mse}")
print(f"R Square is {r2}")
```

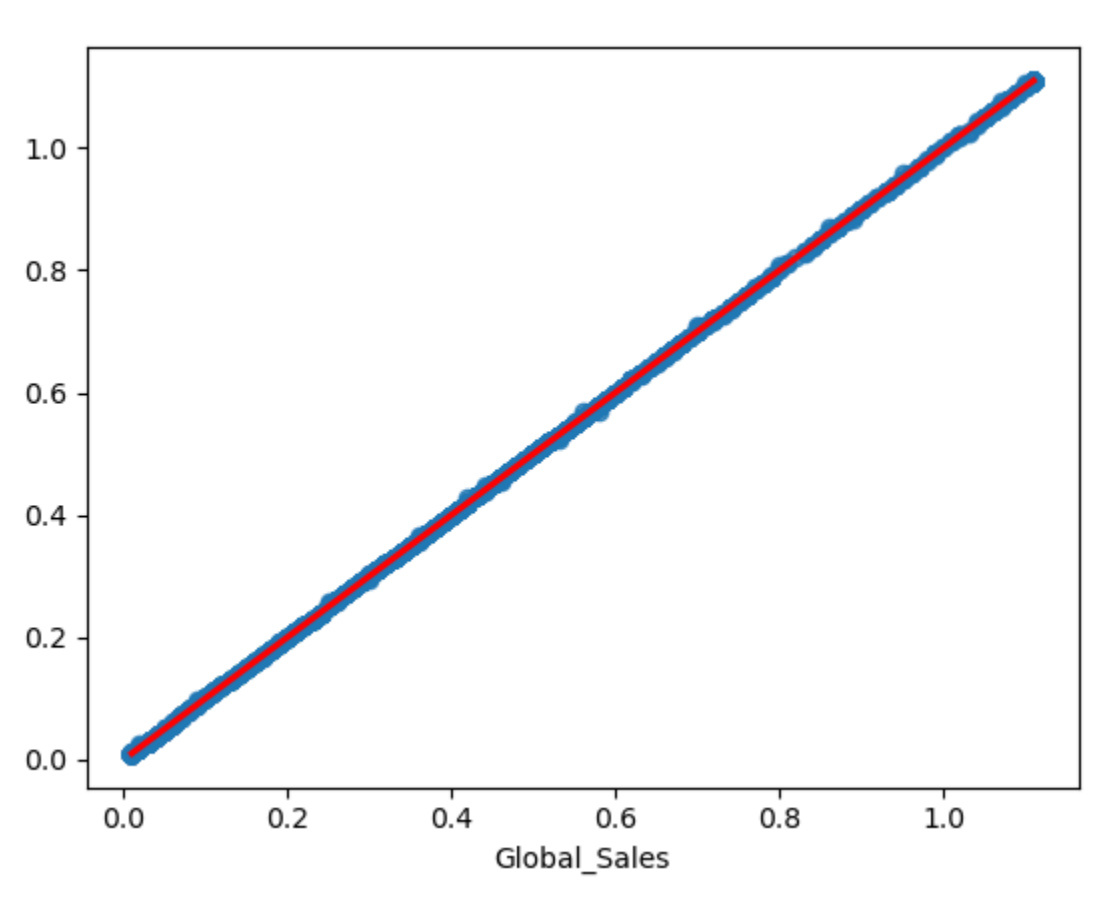
```
Mean Absolute Error is 0.00024240564590365088
Mean Squared Error is 1.1199754525928l6e-06
R Square is 0.9999915226593845
```

Random Forest Regressor

In [174…
```python
#performing random forest regressor
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
rf.fit(X_train,y_train)
rf_y = rf.predict(X_test)
```

In [175…
```python
sns.regplot(x=y_test,y=rf_y,line_kws={"color":"red"})
```

Out[175]:
```
<AxesSubplot:xlabel='Global_Sales'>
```



In [176…
```python
mae = mean_absolute_error(y_test,rf_y)
mse = mean_squared_error(y_test,rf_y)
r2 = r2_score(y_test,rf_y)

print(f"Mean Absolute Error is {mae}")
print(f"Mean Squared Error is {mse}")
print(f"R Square is {r2}")
```

```
Mean Absolute Error is 8.781834918787632e-05
Mean Squared Error is 4.352439398588312e-07
R Square is 0.9999967055428577
```

In [ ]: