Spring 2022

Video game sales data and Analysis

MIS 502 PROJECT REPORT

PROFESSOR JIM RYAN

KUSHAL SHAH

Table of Contents :

Data Description:

This dataset contains a list of video games with sales greater than 100,000 copies. It was generated by a scrape of vg chartz. It was then uploaded to Kaggle as Video Game Sales or vgsales.csv.
Gaming being close to my heart was one of the reasons I decided to go ahead with a dataset related to video games.
The dataset contains 16,598 records. 2 records were dropped due to incomplete information on Kaggle. The script used to scrape the data is based on BeautifulSoup using Python. The dataset has 11 columns or fields, and they are

1. Rank - Ranking of overall sales
2. Name - The games name
3. Platform - Platform of the games release (i.e. PC,PS4, etc.)
4. Year - Year of the game's release
5. Genre - Genre of the game
6. Publisher - Publisher of the game
7. NA_Sales - Sales in North America (in millions)
8. EU_Sales - Sales in Europe (in millions)
9. JP_Sales - Sales in Japan (in millions)
10. Other_Sales - Sales in the rest of the world (in millions)
11. Global_Sales - Total worldwide sales.(in millions)

Potential Outcomes:

We can use the data and analyze various things from running data mining techniques to find cluster analysis, correlation & association and Linear and Random Forest Regression. We can further use data visualization techniques to find different graphs for points like

- The games with the highest sales. The games with the lowest sales
- We can explore the highest sales in which year
- The top games for each platform
- The top genres for games
- The top game Publishers
- The top games in different genres
- The bestselling games in the different regions
- The distribution of sales in different regions

This would help us understand the pattern behind the video game sales and would help to do business related to it.

Data Preparation:

The data needs to be prepared to be used for analysis. It might contain incomplete rows and columns which would lead to errors in the analysis. The Initial dataset on Kaggle mentioned that two records were dropped which would mean there might be more inconsistencies and null values in the data which would need to be addressed. Going through the data, there were some null values in the rows. One such Example was the year 2016. I ran the data through Fillna and dropna commands to help fill up the data and improve the quality of the data. I ran the following code to prepare the data.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
games=pd.read_csv('vgsales.csv')
games = pd.DataFrame(games)
games.columns = [i.lower() for i in games.columns]
games = games.drop_duplicates()
print("Dataset shape :",games.shape)
print(games.head(2))
print(games.tail(2))
games=games.fillna(games.mean)
games.dropna(how = "any",inplace = True)
games.drop("rank", axis=1, inplace=True)
print("Dataset shape after droping row's with null value's :",games.shape)
print(games.head(2))
print(games.tail(2))
```

Outcome of data preparation:

I ran the shape, head and tail for the data before and after the execution of the code. The dataset before the preparation was 16958 by 11 and after the Preparation was 16598 by 10. The head and tail of the data before and after preparation is :

```
Dataset shape : (16598, 11)
   rank              name platform  ...  jp_sales other_sales global_sales
0     1         Wii Sports      Wii  ...      3.77        8.46        82.74
1     2  Super Mario Bros.      NES  ...      6.81        0.77        40.24

[2 rows x 11 columns]
        rank              name platform  ...  jp_sales other_sales global_sales
16596  16599        Know How 2       DS  ...       0.0         0.0         0.01
16597  16600  Spirits & Spells      GBA  ...       0.0         0.0         0.01

[2 rows x 11 columns]
Dataset shape after droping row's with null value's : (16598, 10)
               name platform    year  ... jp_sales other_sales  global_sales
0         Wii Sports      Wii  2006.0  ...     3.77        8.46         82.74
1  Super Mario Bros.      NES  1985.0  ...     6.81        0.77         40.24

[2 rows x 10 columns]
                  name platform    year  ... jp_sales other_sales  global_sales
16596        Know How 2       DS  2010.0  ...      0.0         0.0          0.01
16597  Spirits & Spells      GBA  2003.0  ...      0.0         0.0          0.01

[2 rows x 10 columns]
```

Data Mining techniques

A lot of relevant data can be extracted from the dataset by using different data mining techniques. I ran the numbers in the dataset through different data mining techniques like Cluster analysis, Linear Regression, Random Forest regression and Correlation and Association. I did not think dimensionality reduction would help with this dataset as there were no parts of the data that were not needed. The code and the Outcomes for the Various Data mining techniques are:

Cluster Analysis:

The code I ran for Cluster Analysis was:

```
df3 = data[["na_sales","eu_sales","jp_sales","other_sales","global_sales"]]   # reduced df
df3.fillna(0, inplace=True)
k_groups = KMeans(n_clusters=5, random_state=0).fit(df3)
print(k_groups.labels_)
print(len(k_groups.labels_),df3.shape)
print(k_groups.cluster_centers_)
print(k_groups.cluster_centers_[0])
df3['cluster'] = k_groups.labels_
print(df3.head(3))
print(df3.groupby('cluster').mean())
from sklearn.metrics import silhouette_score
df4 = df3.drop('cluster', axis = 1)
for i in range(3, 10):
    k_groups = KMeans(n_clusters = i).fit(df4)
    labels = k_groups.labels_
    print('K Groups = ', i, 'Silhouette Coeffient = ', silhouette_score(df4, labels))
```

The Outcome of the above code was:

```
[4 2 2 ... 0 0 0]
16598 (16598, 5)
[[1.27194615e-01 6.47218373e-02 4.62271497e-02 2.22807365e-02
  2.60727909e-01]
 [4.21551181e+00 2.84212598e+00 1.11968504e+00 8.55196850e-01
  9.03220472e+00]
 [1.28963636e+01 6.56363636e+00 3.88636364e+00 2.16590909e+00
  2.55131818e+01]
 [1.23937452e+00 7.09675676e-01 2.77274131e-01 2.28115830e-01
  2.45444788e+00]
 [4.14900000e+01 2.90200000e+01 3.77000000e+00 8.46000000e+00
  8.27400000e+01]]
[0.12719461 0.06472184 0.04622715 0.02228074 0.26072791]
   na_sales  eu_sales  jp_sales  other_sales  global_sales  cluster
0     41.49     29.02      3.77         8.46         82.74        4
1     29.08      3.58      6.81         0.77         40.24        2
2     15.85     12.88      3.79         3.31         35.82        2
```

```
cluster
0         0.127519   0.064971  0.046243     0.022332      0.261370
1         4.215512   2.842126  1.119685     0.855197      9.032205
2        12.896364   6.563636  3.886364     2.165909     25.513182
3         1.243336   0.711252  0.278701     0.228950      2.462232
4        41.490000  29.020000  3.770000     8.460000     82.740000
K Groups =  3 Silhouette Coeffient =  0.8669042238815781
K Groups =  4 Silhouette Coeffient =  0.8241613630356502
K Groups =  5 Silhouette Coeffient =  0.7912563318951493
K Groups =  6 Silhouette Coeffient =  0.7275802899015245
K Groups =  7 Silhouette Coeffient =  0.7044658249292596
K Groups =  8 Silhouette Coeffient =  0.6690790967293154
K Groups =  9 Silhouette Coeffient =  0.5945814079849545
```

This meant the 3 Silhouette Coefficient is the most preferred one.

Regression:

I ran Linear Regression and Random Forest Regression on the code and found the R2 score, adjusted R2 score and the MSE for the dataset.

The code for the regression was:

```
def data_encode(x_data):
    for i in x_data.columns:
        x_data[i] = x_data[i].factorize()[0]

    return x_data
x_data = data.drop("global_sales", axis=1)
y_data = data["global_sales"]
x_data = data_encode(x_data)
x_data = data.drop("global_sales",axis = 1)
y_data = data["global_sales"]
x_data = data_encode(x_data)
xtrain,xtest,ytrain,ytest = train_test_split(x_data,y_data)
lr_model = LinearRegression()
lr_model.fit(xtrain,ytrain)
ypred= lr_model.predict(xtest)
n = len(xtest)
p = xtest.shape[1]
r2_value = r2_score(ytest,ypred)
adjusted_r2_score = 1 - (((1-r2_value)*(n-1)) /(n-p-1))
print("r2_score for Linear Reg model : ",r2_score(ytest,ypred))
print("adjusted_r2_score Value       : ",adjusted_r2_score)
print("MSE for Linear Regression     : ",mean_squared_error(ytest,ypred))
rf_model = RandomForestRegressor(n_estimators=200,min_samples_split=20,random_state=43)
rf_model.fit(xtrain,ytrain)
ypred = rf_model.predict(xtest)
n = len(xtest)
p = xtest.shape[1]
r2_value = r2_score(ytest,ypred)
adjusted_r2_score = 1 - (((1-r2_value)*(n-1)) /(n-p-1))
print("r2_score for Random Forest Reg model : ",r2_score(ytest,ypred))
print("adjusted_r2_score Value              : ",adjusted_r2_score)
print("MSE for Random Forest Regression     : ",mean_squared_error(ytest,ypred))
```

The Outcome of the regression was:

```
r2_score for Linear Reg model :  0.17388184832266051
adjusted_r2_score Value       :  0.1728850841435131
MSE for Linear Regression     :  1.3218170450599454
r2_score for Random Forest Reg model :  0.6813740614122379
adjusted_r2_score Value              :  0.6809896189187681
MSE for Random Forest Regression     :  0.509812301991427
```

Correlation and Association:

I finally ran Correlation on the dataset between the different sets of sales to understand it better.

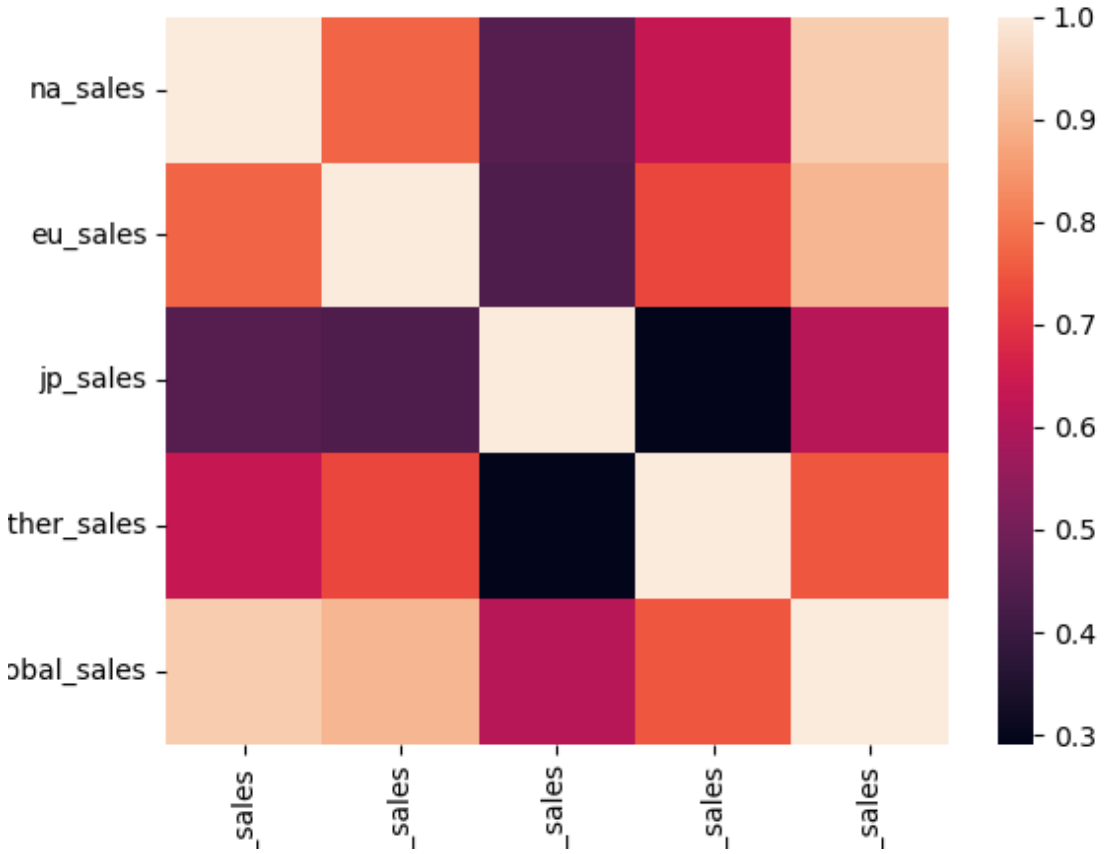
Code used to run Correlation:

```
X = data[["na_sales","eu_sales","jp_sales","other_sales"]]
y = data.global_sales
print(data.corr())
sn.heatmap(data.corr())
plt.show()
print(data.cov())
```

The Outcome of Correlation was:

	na_sales	eu_sales	jp_sales	other_sales	global_sales
na_sales	1.000000	0.767727	0.449787	0.634737	0.941047
eu_sales	0.767727	1.000000	0.435584	0.726385	0.902836
jp_sales	0.449787	0.435584	1.000000	0.290186	0.611816
other_sales	0.634737	0.726385	0.290186	1.000000	0.748331
global_sales	0.941047	0.902836	0.611816	0.748331	1.000000

I also plotted a heatmap to better understand the correlation between the different sales.


na_sales
eu_sales
jp_sales
ther_sales
obal_sales
_sales
_sales
_sales
_sales
_sales
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

Data Visualization techniques:

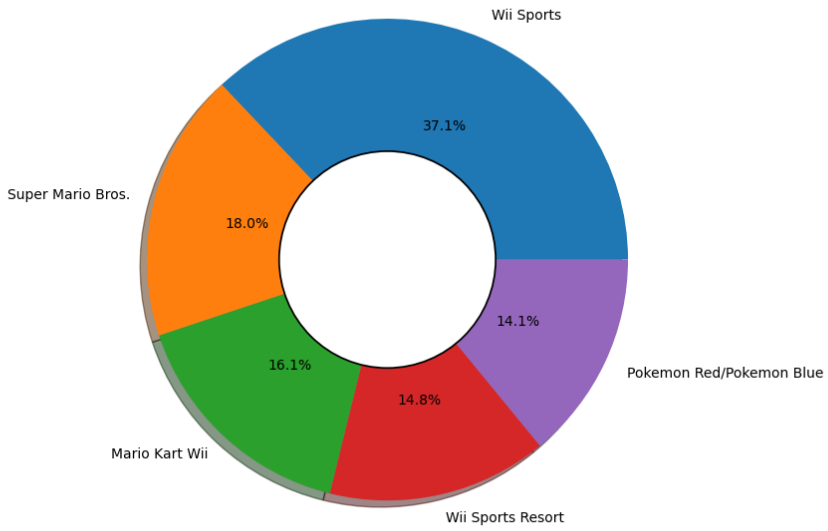
Data Visualization techniques like pie chart, scatter plots, density graphs, bar graphs were used with the help of Matplotlib, Seaborn and plotly.express libraries for the data visualization.

Game with the highest sales

The code used for finding and plotting the games with the highest sales was

```
game = data.loc[:,['Name','Global_Sales']]
game = game.sort_values('Global_Sales', ascending=False)
game = game.head()
fig = plt.figure(figsize=(10,7))
plt.pie(game['Global_Sales'], labels=game['Name'], autopct='%1.1f%%', shadow=True)
centre_circle = plt.Circle((0,0),0.45,color='black', fc='white',linewidth=1.25)
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.axis('equal')
plt.show()
```

The Outcome was a Pie chart which showed us that Wii Sports has the highest global sales.

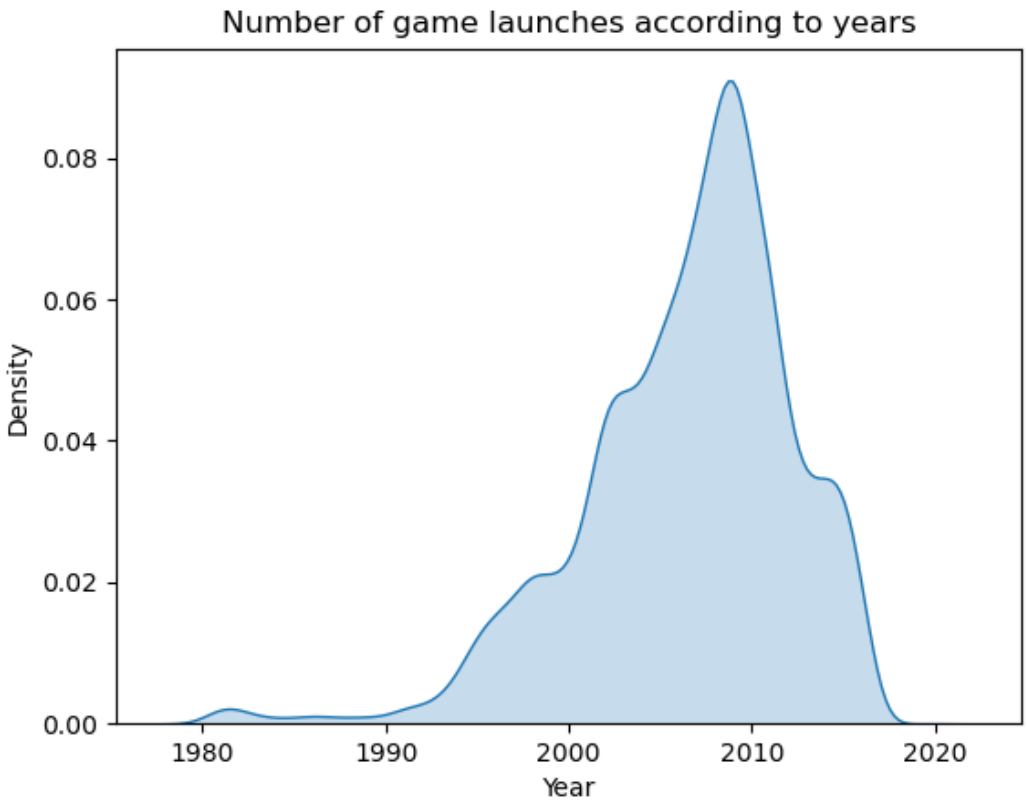


Game launches in different years

The second use of data visualization was to find out the different number of game launches across the years using a density plot. The code used for that was:

```
sn.kdeplot(data=data['Year'], label='Year', shade=True)
plt.title('Number of game launches according to years')
plt.show()
```

The Outcomes of the above code showed that the peak of the highest game launches was between 2005 and 2015

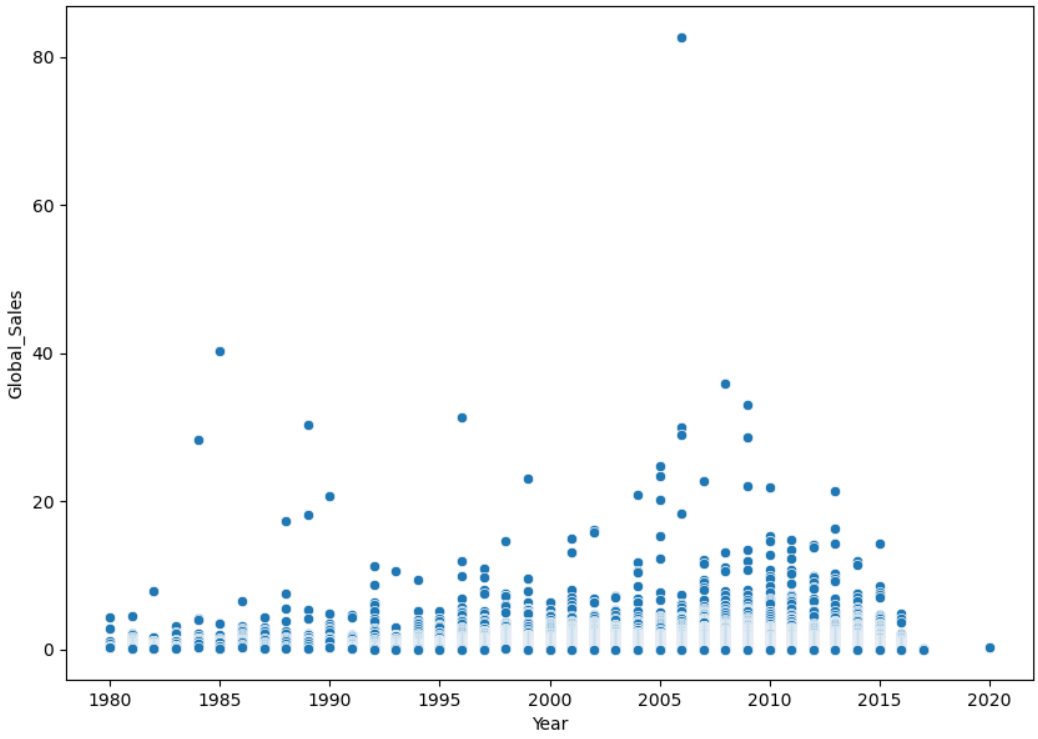


Relation of the global sales with years

For this I analyzed the relation between the global sales and the different years using a scatterplot. The code used for that was.

```
plt.figure(figsize=(10,7))
sn.scatterplot(data=data, x='Year', y='Global_Sales')
plt.show()
```

The outcome of the code was a scatterplot with one Outlier

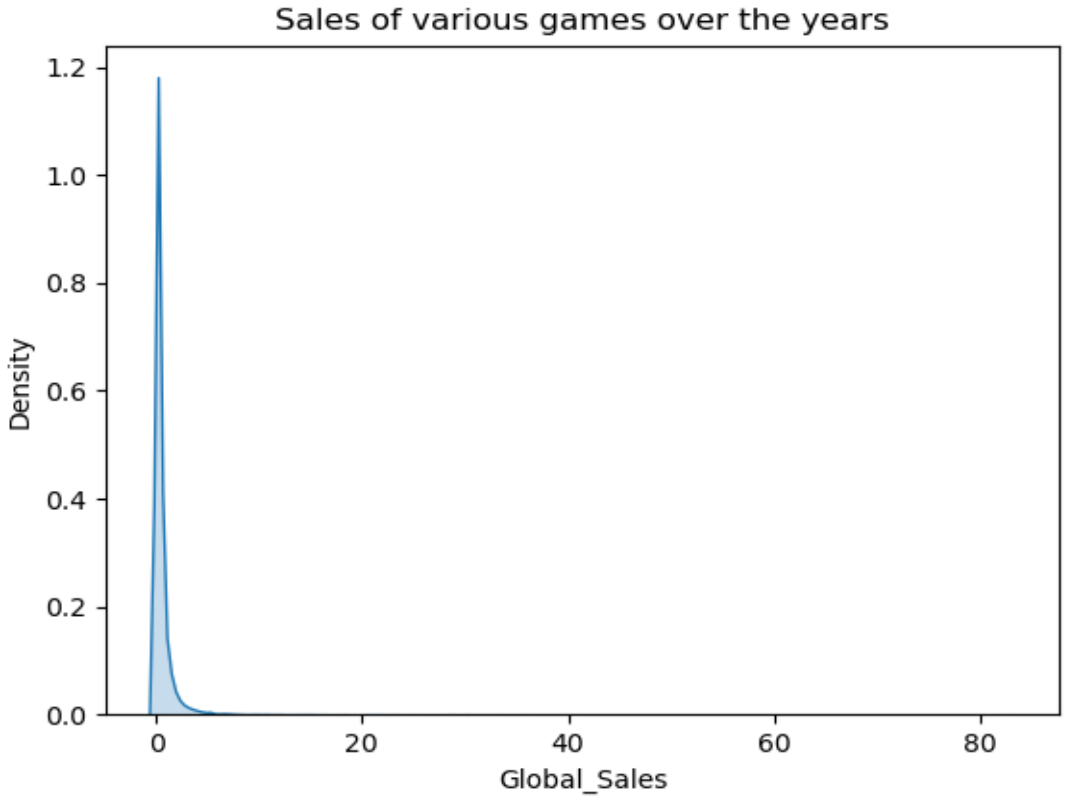


The sales range with most games

I used data visualization to find out the different sales ranges and the range with the most number of games in it using a density plot. The code I used to do the following was:

```
sn.kdeplot(data=data['Global_Sales'], label='Global_Sales', shade=True)
plt.title('Sales of various games over the years')
plt.show()
```

The outcome of the code was a density plot that most games have only a couple of million copies sold overall.



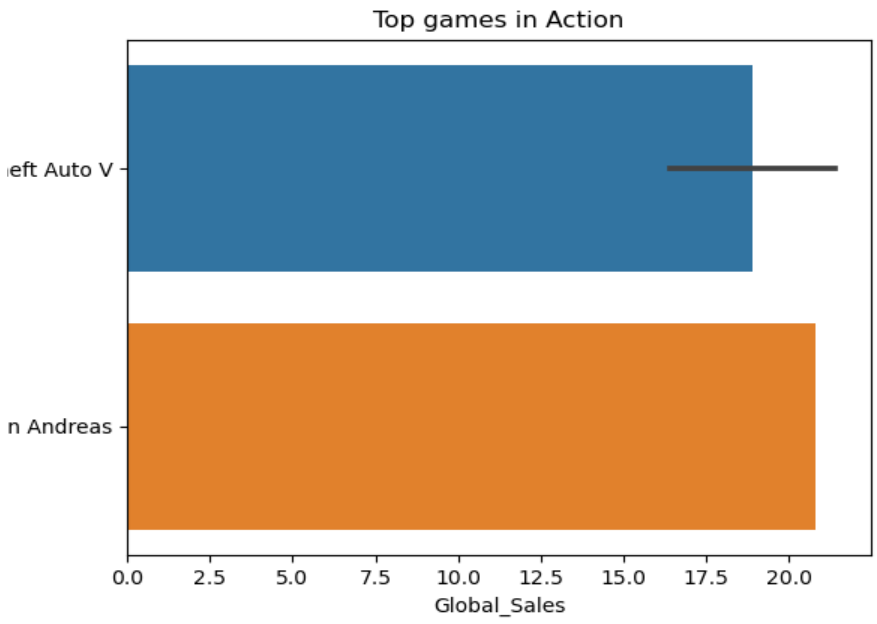
Top games in different genres

I found the top games in different genres using data visualization by bar plot. The code used for that was:

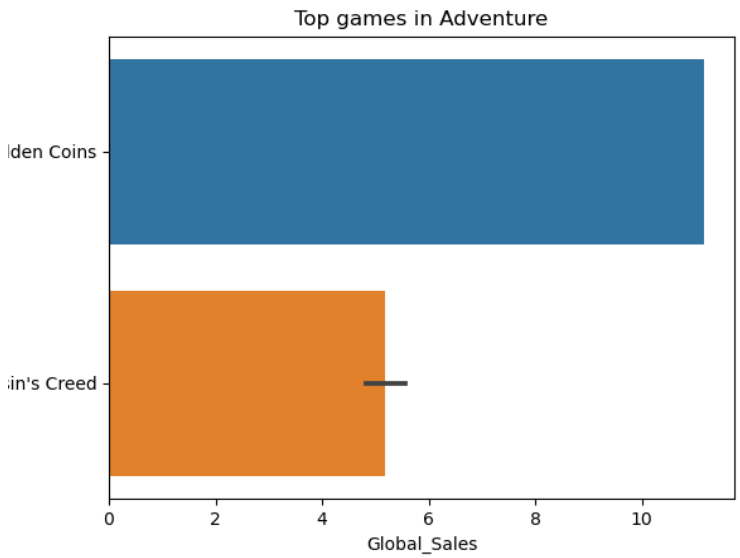
```
for genre,num in zip(genres,range(1,13)):
    df = data[data['Genre']==genre]
    df = df.sort_values('Global_Sales', ascending=False)
    df = df.head(3)
    plt.figure()
    sn.barplot(data=df, x='Global_Sales', y='Name')
    plt.title('Top 5 games in {}'.format(genre))
    plt.show()
```

The outcome of the code gave us a bar plot of the top games of each of the 12 genres in the dataset.

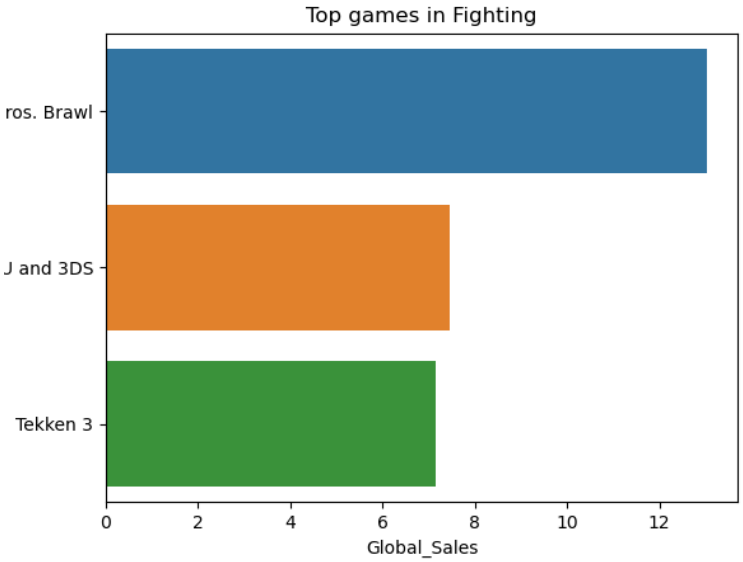
Action



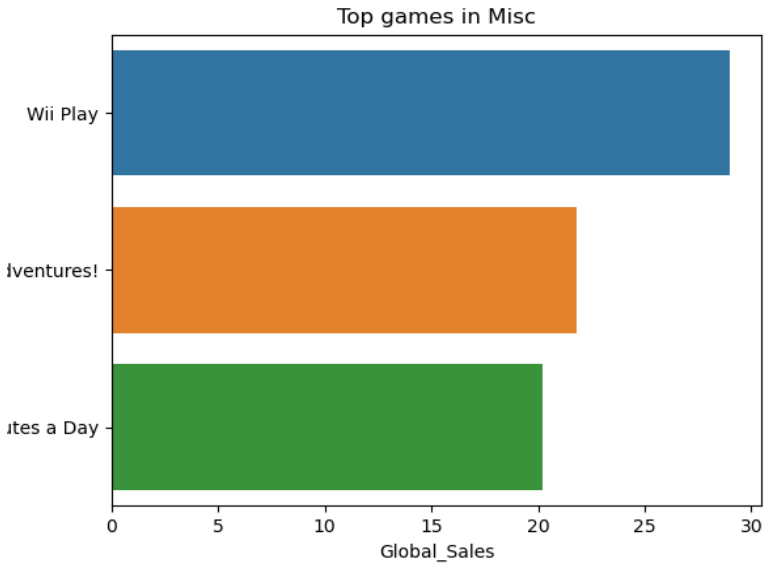
Adventure



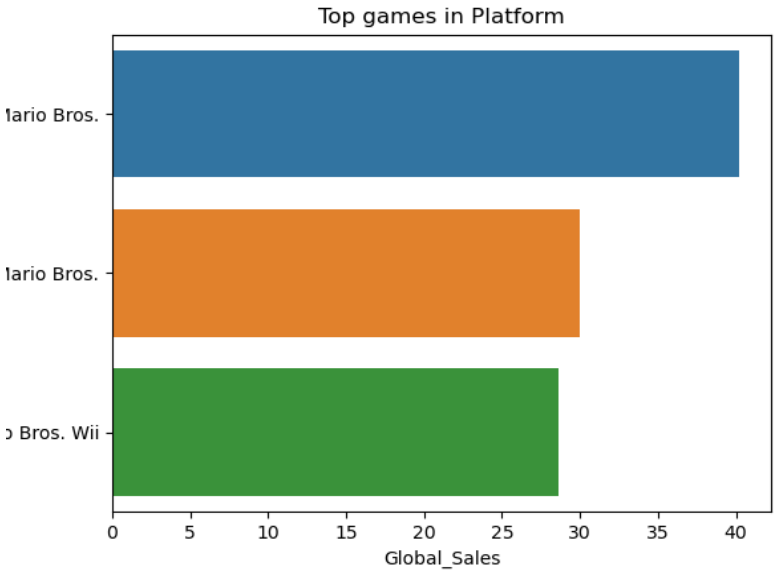
Fighting



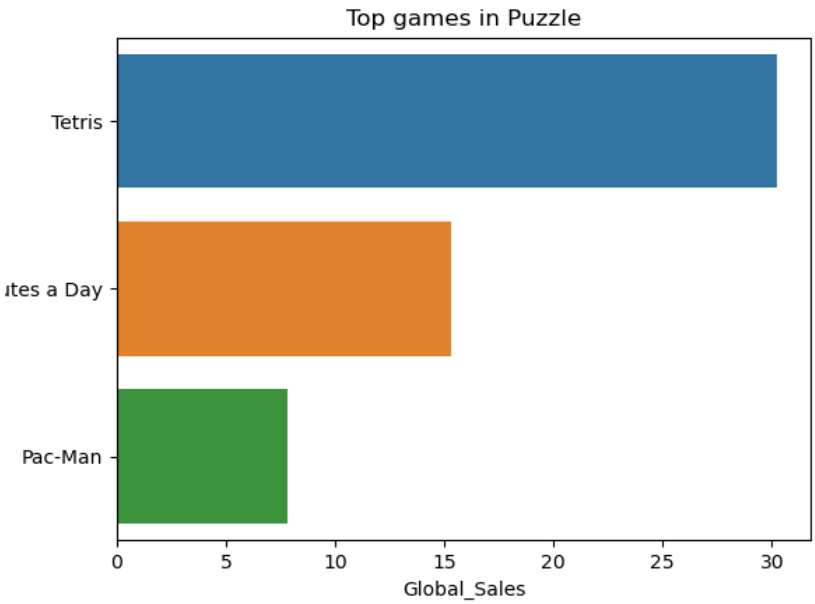
Miscellaneous



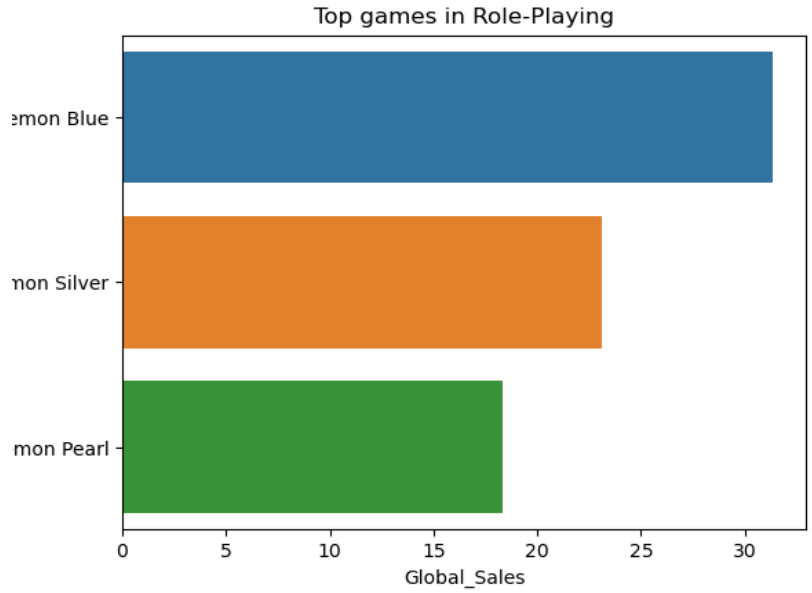
Platform



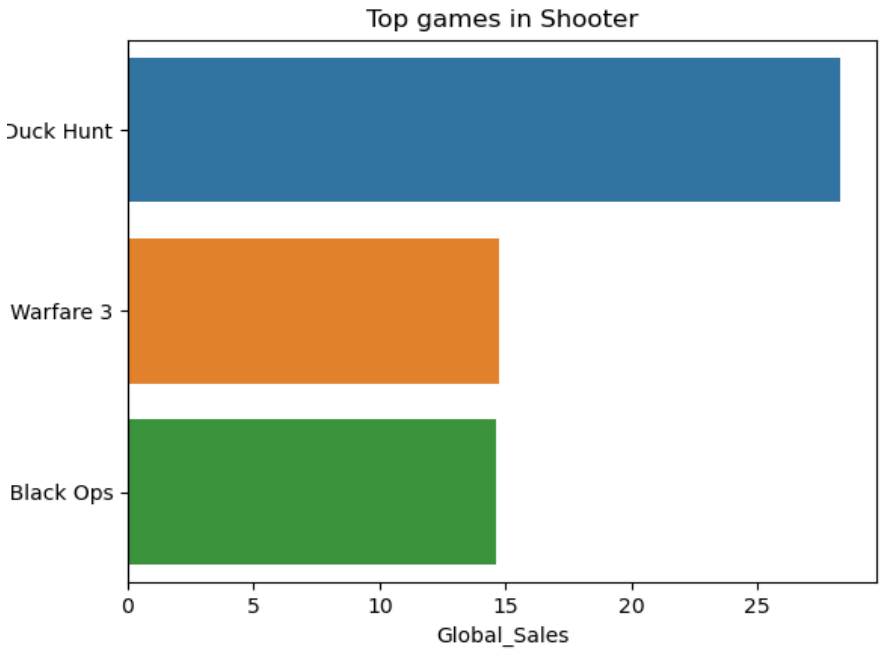
Puzzle



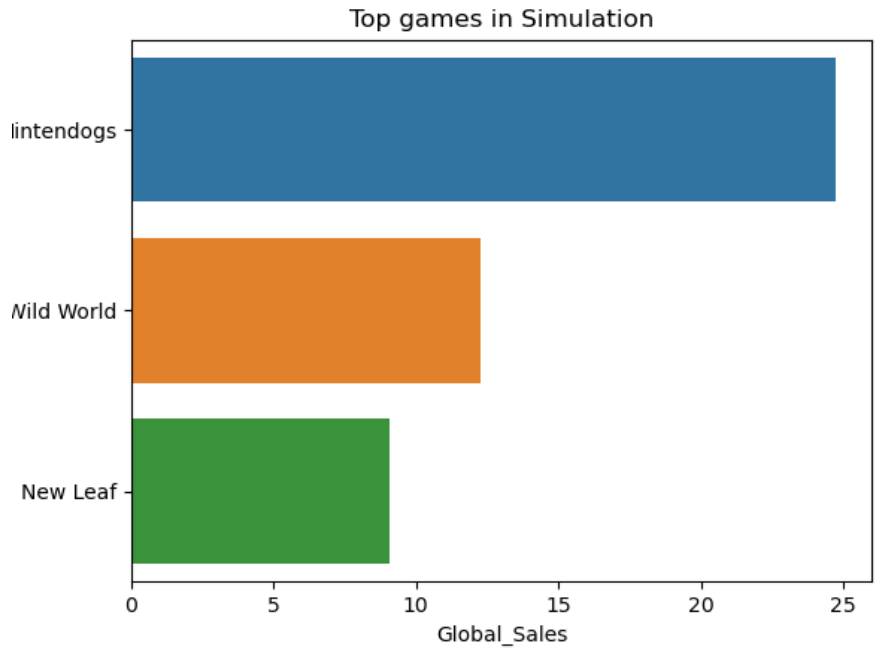
Role-playing



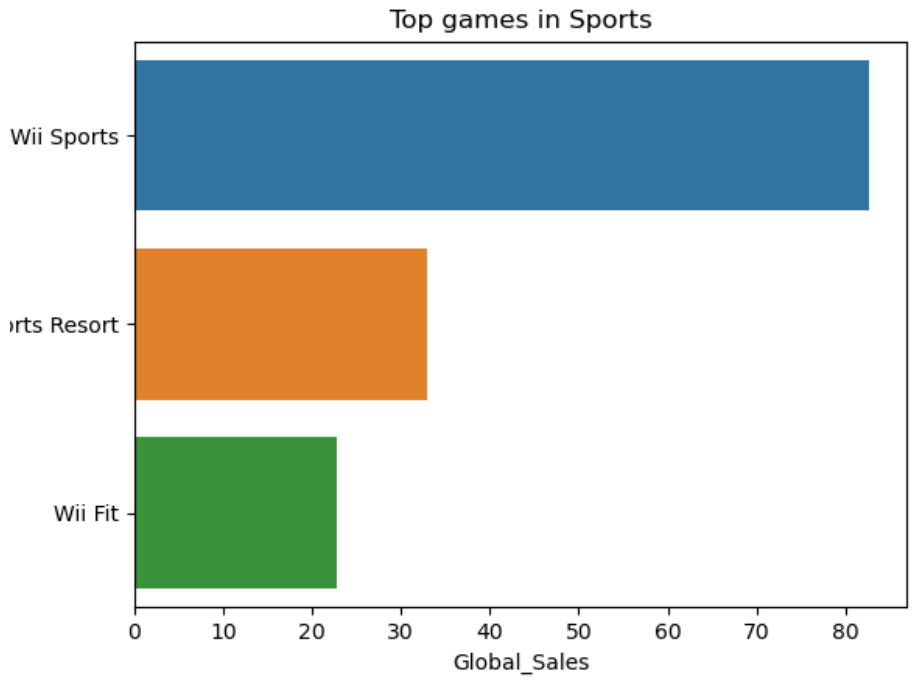
Shooter



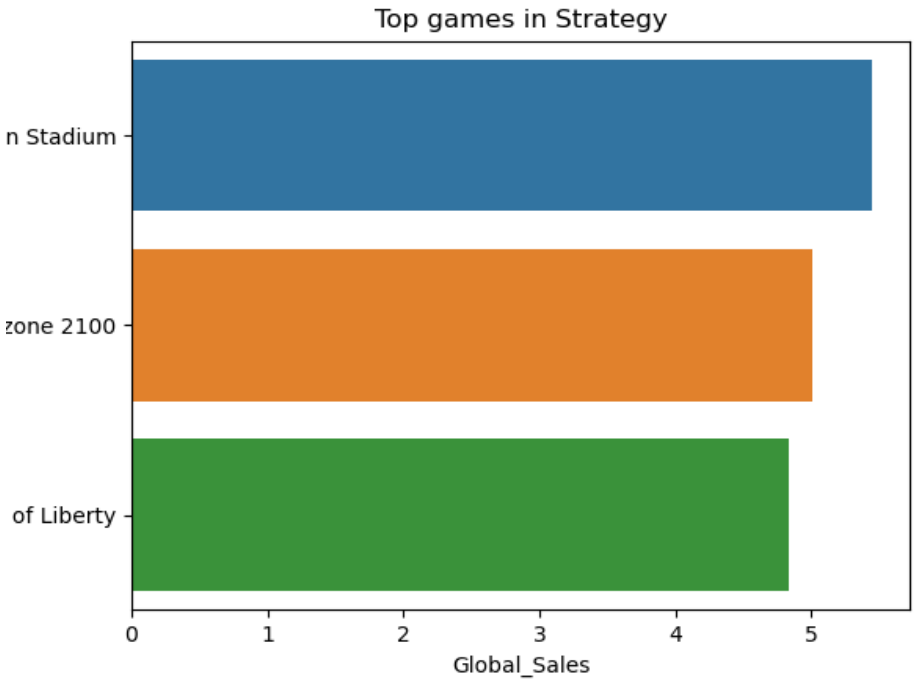
Simulation



Sports



Strategy

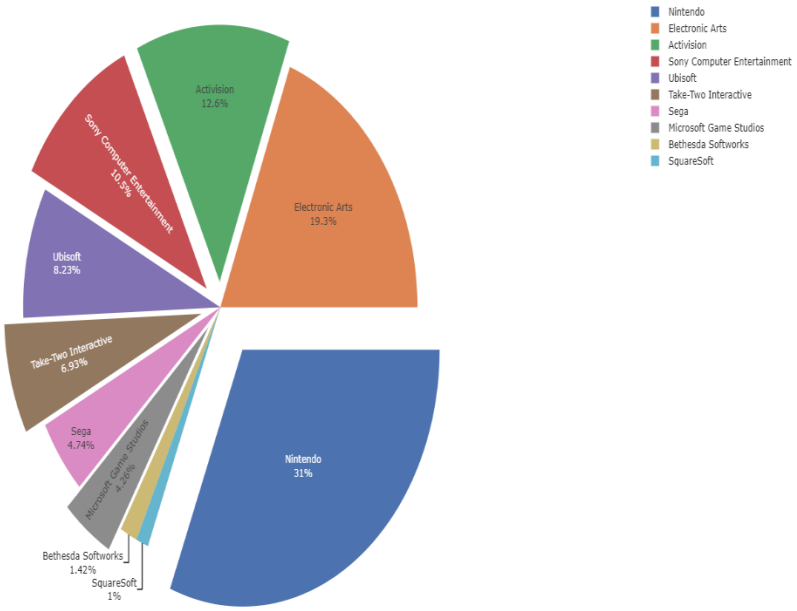


Top Publisher in terms of sales

I used data visualization using a pie chart to find out the top publisher in terms of sales. The code used for that was:

```
fig = px.pie(publisher, names='Publisher', values='total_sales', template='seaborn')
fig.update_traces(rotation=90, pull=[0.2,0.1,0.1,0.1,0.1], textinfo="percent+label")
fig.show()
```

The result of running the above code gave us the top publisher in terms of sale as Nintendo.



Citations:

Dataset link :- https://www.kaggle.com/datasets/gregorut/videogamesales

Script to scrap data: https://github.com/GregorUT/vgchartzScrape.

Website for the data: vgchartz.com