Practical – 1

Aim: Study of Python Basic Libraries like a Numpy. Perform the following task using Numpy library.

- Creating blank array, with predefined data, with pattern specific data
- Slicing and Updating elements,
- Shape manipulations
- Looping over arrays.

- **Code:**

```
import numpy as np
a = np.array([10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
print(a)
print(f"Slicing array: {a[5:]}")
print(f"Slicing array: {a[:5]}")
print(f"Slicing array: {a[3:8]}")
a[5] = 999
a[2] = 888
print(a)
b = a.reshape(2, 5)
print(f"reshaping into 2x5: \n{b}")
c = a.reshape(5, 2)
print(f"reshaping into 5x2: \n{c}")
print("\nLooping over Array:\n")
for i in a:
  print(i)
for i in b:
  print(i)
```

- **Output:**

```
[ 10  20  30  40  50  60  70  80  90 100]
Slicing array: [ 60  70  80  90 100]
Slicing array: [10 20 30 40 50]
Slicing array: [40 50 60 70 80]
[ 10  20 888  40  50 999  70  80  90 100]
reshaping into 2x5:
[[ 10  20 888  40  50]
 [999  70  80  90 100]]
reshaping into 5x2:
[[ 10  20]
 [888  40]
 [ 50 999]
 [ 70  80]
 [ 90 100]]
```

```
Looping over Array:

10
20
888
40
50
999
70
80
90
100
[ 10  20 888  40  50]
[999  70  80  90 100]
```

Practical – 2

Aim: Study of Python Libraries for ML application such as Pandas and Matplotlib Perform the following task using Pandas & Matplotlib library.

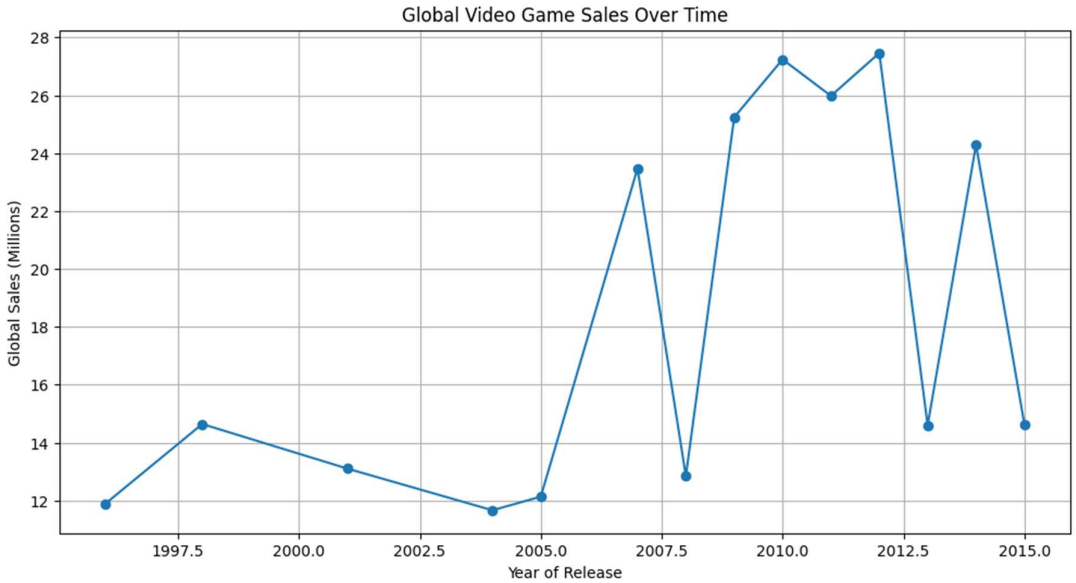
- Creating data frame
- Reading files
- Slicing manipulations
- Exporting data to files
- Columns and row manipulations with loops
- Use pandas for masking data and reading in Boolean format.
- Importing matplotlib
- Simple line chart
- Correlation chart
- Histogram
- Plotting of Multivariate data
- Plot Pi Chart

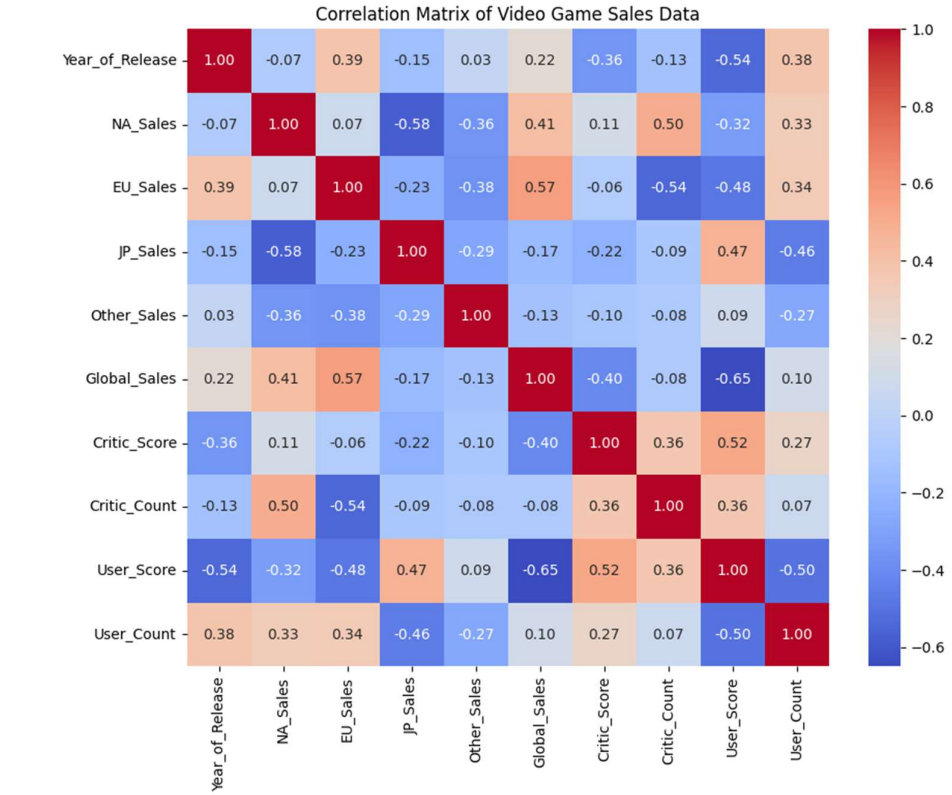
- **Code:**

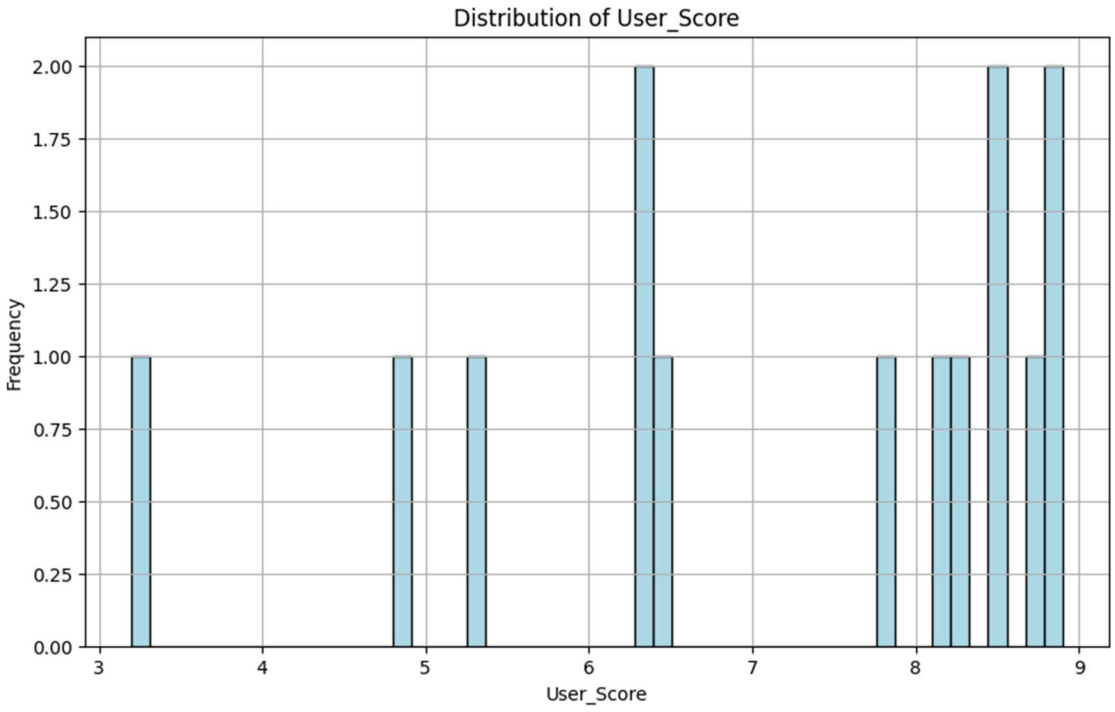
```
import kagglehub
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

path = kagglehub.dataset_download("sidtwr/videogames-sales-dataset")
# print("Path to dataset files:", path)
df = pd.read_csv("/kaggle/input/videogames-sales-
dataset/Video_Games_Sales_as_at_22_Dec_2016.csv")
# df.head()
temp = df.iloc[30:50]
# print(temp)
# temp.to_csv('temp.csv', index=False)
# from google.colab import files
# files.download('temp.csv')
colManipulation = df.iloc[50:60:5]
# print(colManipulation)
sales_by_year =
temp.groupby('Year_of_Release')['Global_Sales'].sum().reset_index()
sales_by_year['Year_of_Release'] = pd.to_numeric(sales_by_year['Year_of_Release'],
errors='coerce')
sales_by_year.dropna(subset=['Year_of_Release'], inplace=True)
sales_by_year.sort_values('Year_of_Release', inplace=True)
#line chart
plt.figure(figsize=(12, 6))
plt.plot(sales_by_year['Year_of_Release'], sales_by_year['Global_Sales'], marker='o',
linestyle='-')
plt.title('Global Video Game Sales Over Time')
plt.xlabel('Year of Release')
plt.ylabel('Global Sales (Millions)')
plt.grid(True)
plt.show()
```

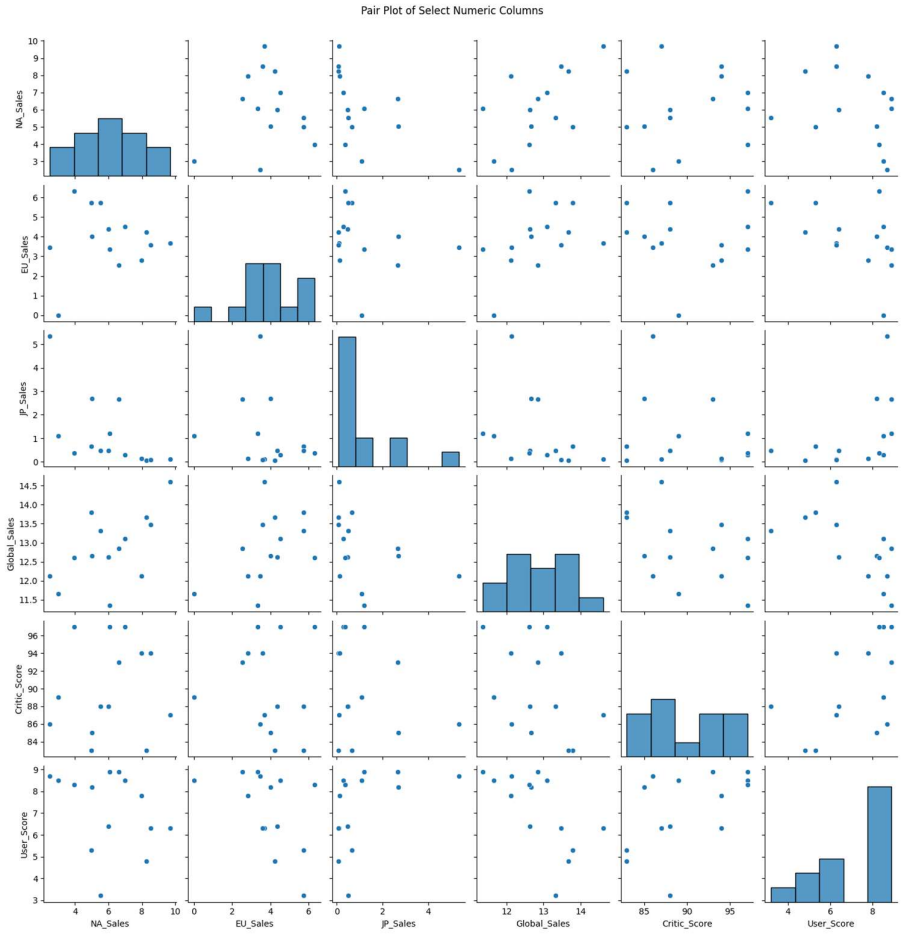
```
#corelation chart
numeric_cols = temp.select_dtypes(include=['float64', 'int64'])
correlation_matrix = numeric_cols.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Video Game Sales Data')
plt.show()
#histogram
plt.figure(figsize=(10, 6))
plt.hist(temp['User_Score'], bins=50, color='lightblue', edgecolor='black')
plt.title('Distribution of User_Score')
plt.xlabel('User_Score')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
#multivariant data
cols_for_pairplot = ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Global_Sales', 'Critic_Score',
'User_Score']
sns.pairplot(temp[cols_for_pairplot].dropna())
plt.suptitle('Pair Plot of Select Numeric Columns', y=1.02)
plt.show()
games_per_platform = temp['Platform'].value_counts().reset_index()
games_per_platform.columns = ['Platform', 'Number_of_Games']
top_platforms = games_per_platform.head(10)
# pie chart
plt.figure(figsize=(10, 10))
plt.pie(top_platforms['Number_of_Games'], labels=top_platforms['Platform'],
autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Game Releases by Platform (Top 10)')
plt.axis('equal')
plt.show()
```

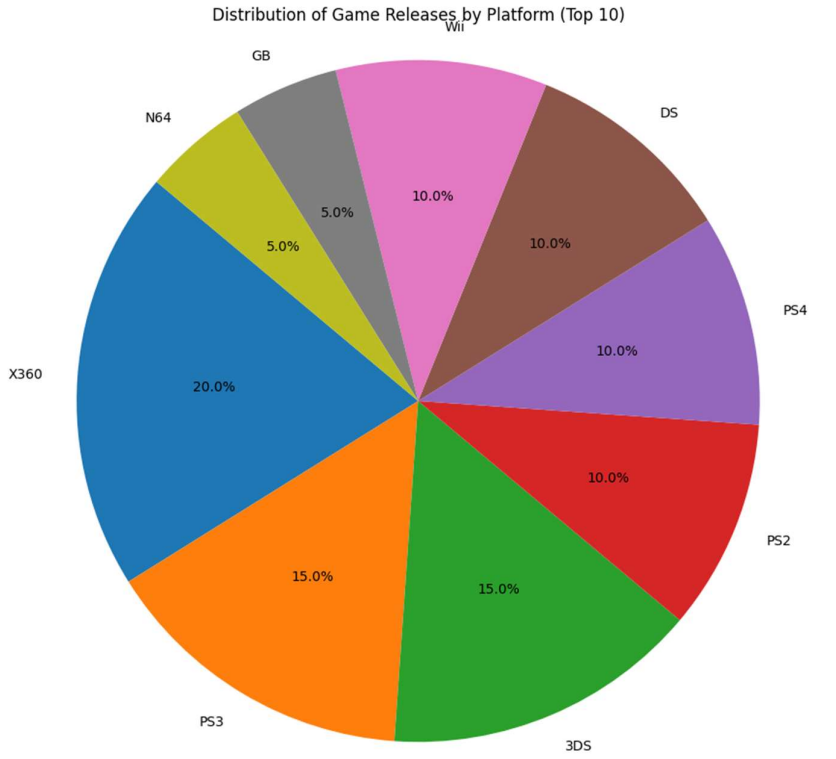
- **Output:**




Correlation Matrix of Video Game Sales Data
Year_of_Release 1.00 -0.07 0.39 -0.15 0.03 0.22 -0.36 -0.13 -0.54 0.38
NA_Sales -0.07 1.00 0.07 -0.58 -0.36 0.41 0.11 0.50 -0.32 0.33
EU_Sales 0.39 0.07 1.00 -0.23 -0.38 0.57 -0.06 -0.54 -0.48 0.34
JP_Sales -0.15 -0.58 -0.23 1.00 -0.29 -0.17 -0.22 -0.09 0.47 -0.46
Other_Sales 0.03 -0.36 -0.38 -0.29 1.00 -0.13 -0.10 -0.08 0.09 -0.27
Global_Sales 0.22 0.41 0.57 -0.17 -0.13 1.00 -0.40 -0.08 -0.65 0.10
Critic_Score -0.36 0.11 -0.06 -0.22 -0.10 -0.40 1.00 0.36 0.52 0.27
Critic_Count -0.13 0.50 -0.54 -0.09 -0.08 -0.08 0.36 1.00 0.36 0.07
User_Score -0.54 -0.32 -0.48 0.47 0.09 -0.65 0.52 0.36 1.00 -0.50
User_Count 0.38 0.33 0.34 -0.46 -0.27 0.10 0.27 0.07 -0.50 1.00
Year_of_Release
NA_Sales
EU_Sales
JP_Sales
Other_Sales
Global_Sales
Critic_Score
Critic_Count
User_Score
User_Count
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


Distribution of User_Score
Frequency
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
3
4
5
6
7
8
9
User_Score


Pair Plot of Select Numeric Columns
NA_Sales
EU_Sales
JP_Sales
Global_Sales
Critic_Score
User_Score


Distribution of Game Releases by Platform (Top 10)
Wii
DS
PS4
PS2
3DS
PS3
X360
N64
GB
10.0%
10.0%
10.0%
10.0%
15.0%
15.0%
20.0%
5.0%
5.0%

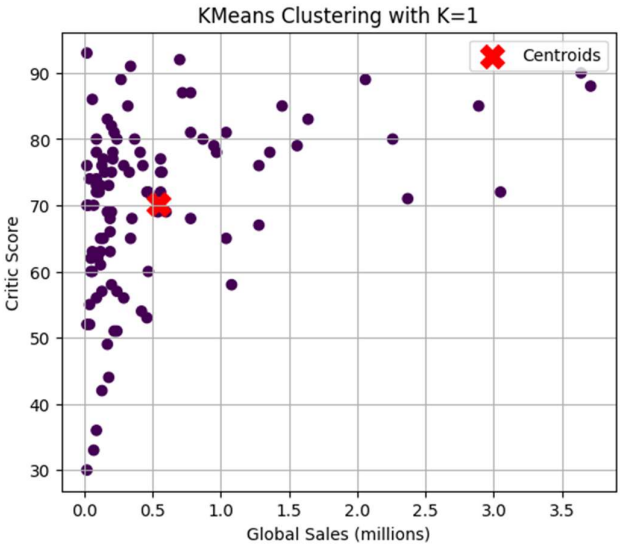
Practical – 8

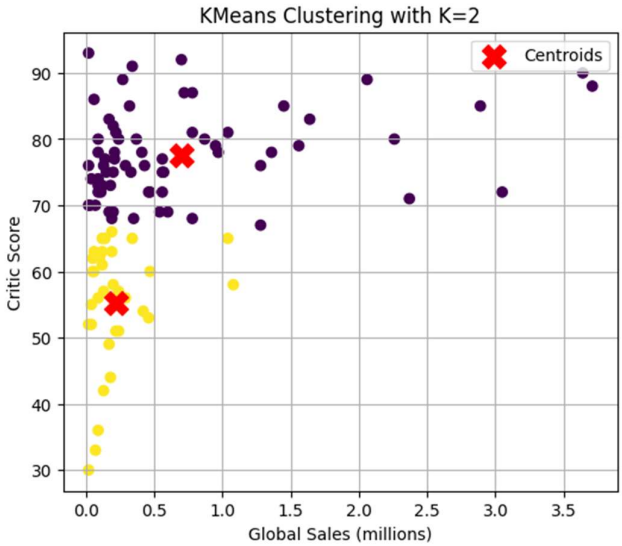
Aim: Write a program to implement K-mean clustering in python.

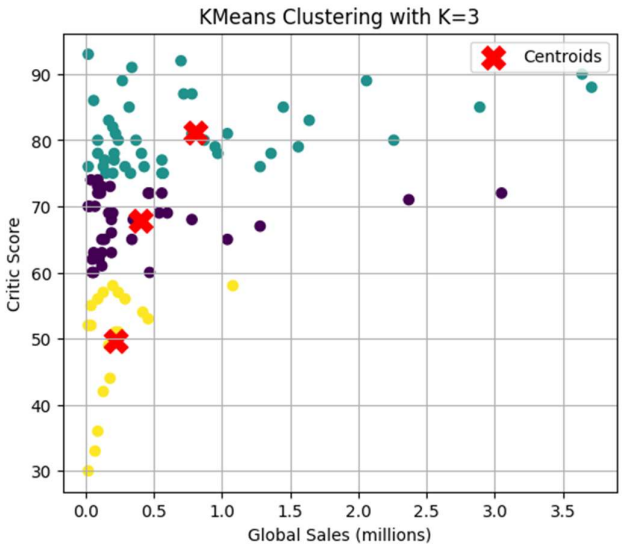
- **Code:**

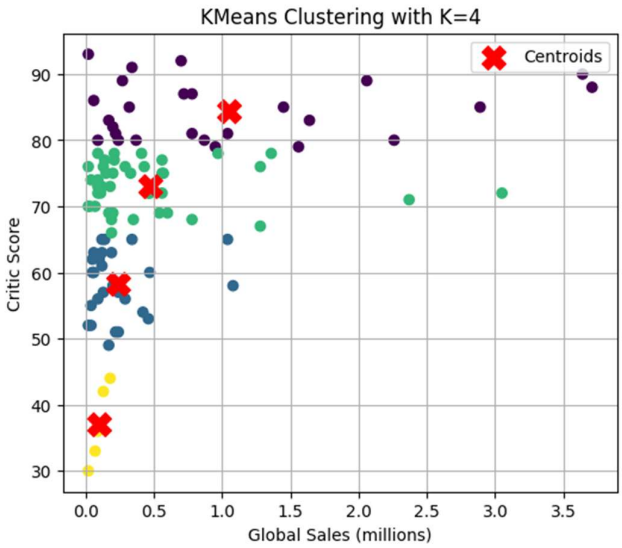
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import numpy as np

df = pd.read_csv("/kaggle/input/videogames-sales-
dataset/Video_Games_Sales_as_at_22_Dec_2016.csv")
df = df[['Global_Sales', 'Critic_Score']].dropna()
df_sample = df.sample(n=100, random_state=42)
X = df_sample[['Global_Sales', 'Critic_Score']]
# Loop through K = 1 to 5
for k in range(1, 6):
    kmeans = KMeans(n_clusters=k, init='k-means++', n_init=10, random_state=42)
    kmeans.fit(X)
    centroids = kmeans.cluster_centers_
    labels = kmeans.labels_
    plt.figure(figsize=(6, 5))
    plt.scatter(X['Global_Sales'], X['Critic_Score'], c=labels, cmap='viridis')
    plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='X', s=200,
label='Centroids')
    plt.title(f"KMeans Clustering with K={k}")
    plt.xlabel("Global Sales (millions)")
    plt.ylabel("Critic Score")
    plt.legend()
    plt.grid(True)
    plt.show()
```

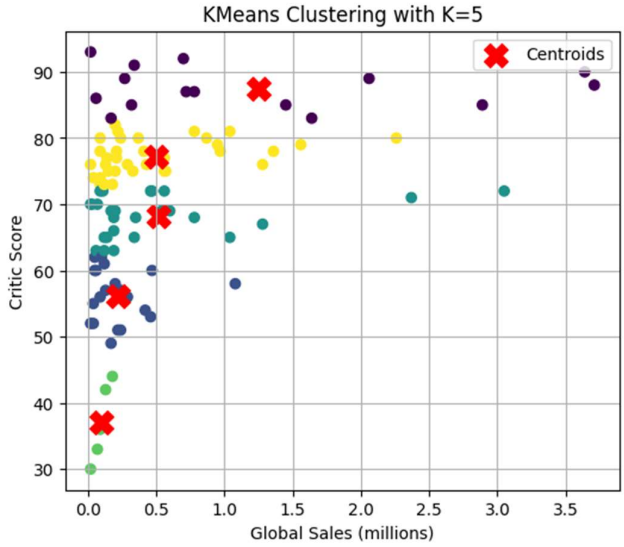
- **Output:**






KMeans Clustering with K=3
Centroids
Critic Score
Global Sales (millions)
90
80
70
60
50
40
30
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5


KMeans Clustering with K=4
Centroids
Critic Score
Global Sales (millions)
90
80
70
60
50
40
30
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5


KMeans Clustering with K=5
Centroids
Critic Score
Global Sales (millions)
90
80
70
60
50
40
30
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5