Clustering Analysis

Global development measurement

Clustering:

A Comprehensive Overview

Clustering analysis is a technique used in machine learning and data analysis to group similar data points into clusters based on their characteristics or features. The goal of clustering is to discover hidden patterns or structures within the data and to organize it in a way that reveals meaningful relationships.

"Clustering is the unsupervised way of finding structure in data"

Business Objective of clustering on World Development Measures :

- Identifying Development Profiles
- Investment Prioritization
- Global Market Segmentation
- Policy Customization
- Public Policy and Diplomacy
- Monitoring Sustainable Development Goals (SDGs)

Data Set:

- Dataset contains 2704 rows and 25 columns.

- Columns present in data are: Birth Rate, Business Tax Rate, CO2 Emissions, Country,Days to Start Business,Ease of Business,Energy Usage,GDP,Health Exp % GDP,Health Exp/Capita,Hours to do Tax,Infant Mortality Rate Internet Usage,Lending Interest,Life Expectancy Female,Life Expectancy Male,Mobile Phone Usage,Number of Records,Population 0-14,Population 15-64,Population 65+,Population Total', 'Population Urban,Tourism Inbound,Tourism Outbound'

```
#Importing the dataset

data=pd.read_excel('/content/World_development_mesurement (1).xlsx')
data
```

	Birth Rate	Business Tax Rate	CO2 Emissions	Country	Days to Start Business	Ease of Business	Energy Usage	GDP	Health Exp % GDP	Exp
0	0.020	NaN	87931.0	Algeria	NaN	NaN	26998.0	$54,790,058,957	0.035	
1	0.050	NaN	9542.0	Angola	NaN	NaN	7499.0	$9,129,594,819	0.034	
2	0.043	NaN	1617.0	Benin	NaN	NaN	1983.0	$2,359,122,303	0.043	
3	0.027	NaN	4276.0	Botswana	NaN	NaN	1836.0	$5,788,311,645	0.047	
4	0.046	NaN	1041.0	Burkina Faso	NaN	NaN	NaN	$2,610,959,139	0.051	
...	...	...	...	...	...	...	...	...	...	
2699	NaN	NaN	NaN	Turks and Caicos Islands	NaN	NaN	NaN	NaN	NaN	
2700	0.013	46.4%	NaN	United States	5.0	4.0	2132446.0	$16,244,600,000,000	0.179	
2701	0.015	41.9%	NaN	Uruguay	7.0	85.0	NaN	$50,004,354,667	0.089	
2702	0.020	61.9%	NaN	Venezuela, RB	144.0	180.0	NaN	$381,286,223,859	0.046	
2703	0.011	NaN	NaN	Virgin Islands (U.S.)	NaN	NaN	NaN	NaN	NaN	

2704 rows × 25 columns

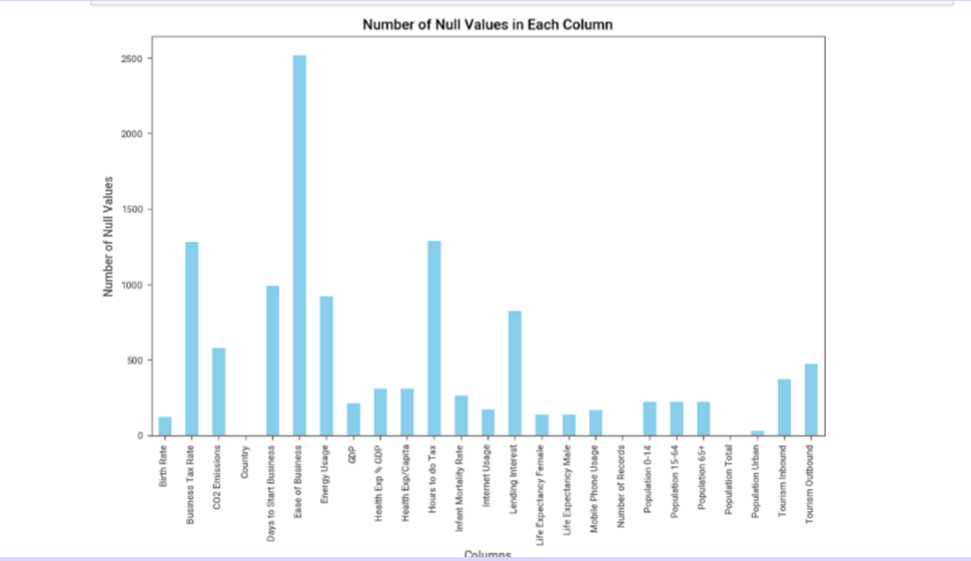
Exploratory Data Analysis

```
[ ] data2['Country'].nunique()

    208
```

```
#Grouping data with respect to country column
grouped_data=data2.groupby('Country')

#getting the count of rows in each group
count_data=grouped_data.size()
count_data
```

```
Country
Afghanistan              13
Albania                  13
Algeria                  13
American Samoa           13
Andorra                  13
                         ..
Vietnam                  13
Virgin Islands (U.S.)    13
Yemen, Rep.              13
Zambia                   13
Zimbabwe                 13
Length: 208, dtype: int64
```

Null values detection and treatement



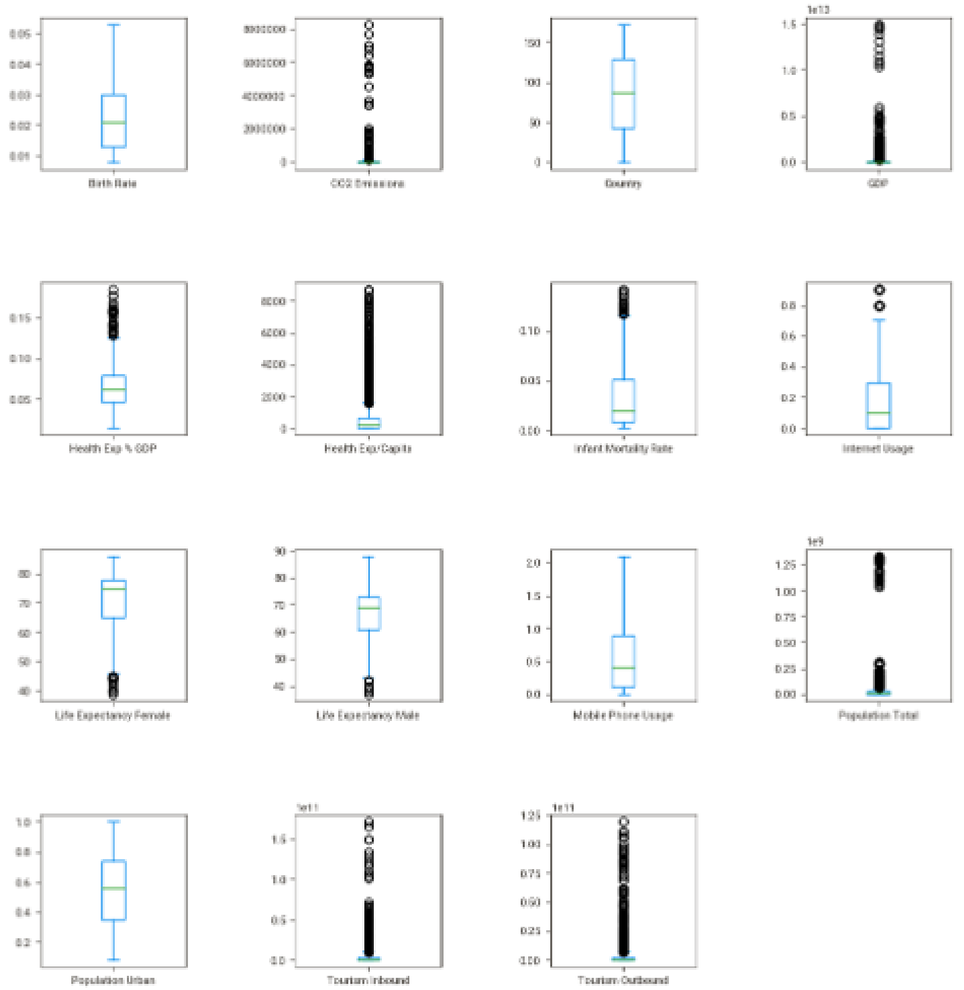

```
# List of columns for which null values need to be imputed
columns_to_impute = ['Birth Rate', 'CO2 Emissions', 'GDP', 'Health Exp % GDP',
                     'Health Exp/Capita', 'Infant Mortality Rate', 'Internet Usage',
                     'Life Expectancy Female', 'Life Expectancy Male', 'Mobile Phone Usage',
                     'Population Total', 'Population Urban', 'Tourism Inbound',
                     'Tourism Outbound']

# Iterate through each column for imputation
for column in columns_to_impute:
    # Group the data by 'Country' and calculate the median for each country
    median_per_country = data2.groupby('Country')[column].median().reset_index()

    # Merge the original DataFrame with the median data
    data2_imputed = pd.merge(data2, median_per_country, on='Country', suffixes=('', '_median'))

    # Fill NaN values in the current column with the median for each country
    data2_imputed[column].fillna(data2_imputed[column + '_median'], inplace=True)

    # Drop the auxiliary column
    data2_imputed.drop(column + '_median', axis=1, inplace=True)

# Display the result after imputation
#print(data2_imputed)
```

```
imputed' is the DataFrame after imputation

f columns to check for null values
to_check_null = ['Birth Rate', 'CO2 Emissions', 'GDP', 'Health Exp % GDP',
                 'Health Exp/Capita', 'Infant Mortality Rate', 'Internet Usage',
                 'Life Expectancy Female', 'Life Expectancy Male', 'Mobile Phone Usage',
                 'Population Total', 'Population Urban', 'Tourism Inbound',
                 'Tourism Outbound']

e through each column for checking null values and removing rows
mn_to_check in columns_to_check_null:
eck for countries with null values in the specified column
tries_with_null_values = data2_imputed[data2_imputed[column_to_check].isna()]['Country'].unique()
nt(f"Countries with Null {column_to_check}:")
nt(countries_with_null_values)

move rows for the specified countries where the specified column is null
2_imputed = data2_imputed[~((data2_imputed['Country'].isin(countries_with_null_values)) & data2_imputed[column_to_check].isna())]

 the resulting DataFrame to 'data3'
data2_imputed

y the result after removing null values
ata3.sample(5))
```

To address extensive null values, we initially excluded columns with over 30% nulls. Considering the data's country focus, we imputed missing values for each column using the country-specific median. If any columns still had nulls, we opted to remove the corresponding country to ensure data integrity for subsequent analysis.

Outliers Detection

```
In [35]: data_f.plot(kind='box',layout=(7,4), subplots=True,sharey=False,figsize=(12,23))
         plt.subplots_adjust(hspace=0.75, wspace=0.75)
```



Removing Outliers

```
]: # Finding Outliers for the Dataset

   # Function to remove outliers based on IQR
   def remove_outliers_iqr(data_f, threshold=1.5):
       Q1 = data_f.quantile(0.25)
       Q3 = data_f.quantile(0.75)
       IQR = Q3 - Q1
       lower_bound = Q1 - threshold * IQR
       upper_bound = Q3 + threshold * IQR
       return data_f[((data_f >= lower_bound) & (data_f <= upper_bound)).all(axis=1)]
```

```
]: # Removing outliers
   data_no_outliers = remove_outliers_iqr(data_f)

   # Displaying the DataFrame without outliers
   print("Original DataFrame:")
   print(data_f)
   print("\nDataFrame without outliers:")
   print(data_no_outliers)
```

After removing outliers dimension of data is:

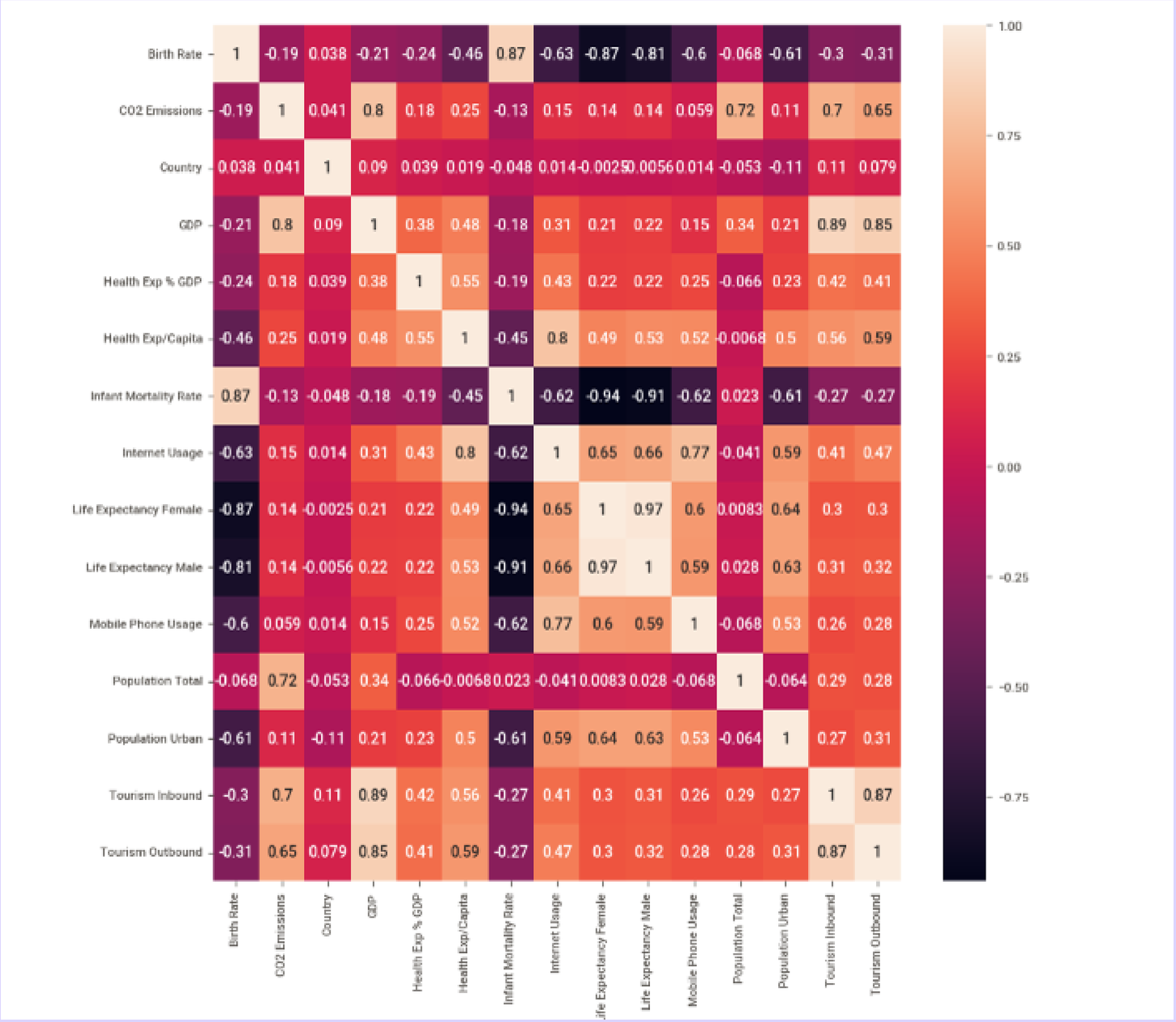
[1178 rows x 15 columns]

As we lose over 50% of data if we remove outliers, we are proceeding without removing them.

Correlation Analysis:

Correlation:-
Highly Correlated columns are:-

- Birth Rate --- Infant Mortality Rate (0.871011)
- Birth Rate --- Life Expectancy Female (-0.865798)
- Birth Rate --- Life Expectancy Male (-0.818383)
- Infant Mortality Rate -- Life Expectancy Female (-0.938558)
- Infant Mortality Rate -- Life Expectancy Male (-0.913536)
- Life Expectancy Female---Life Expectancy Male (0.975040)
- CO2 Emissions --- GDP (0.816268)
- GDP --- Tourism Inbound (0.888174)
- GDP --- Tourism Outbound (0.858800)
- Tourism Inbound --- Tourism Outbound (0.862200)



Clustering Algorithms:

1

Hierarchical Clustering:

2

K-Means Clustering:

3

DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

01

Hierarchical Clustering:

- Hierarchical clustering builds a tree-like hierarchy of clusters.
- It can be agglomerative (bottom-up) or divisive (top-down).
- Agglomerative starts with individual data points and merges them into clusters, while divisive starts with one cluster and splits it recursively.

02

K-Means Clustering:

- K-Means is one of the most popular and widely used clustering algorithms.
- It partitions the data into K clusters based on similarity.
- Each cluster is represented by its centroid, and data points are assigned to the cluster whose centroid is closest to them.

03

DBSCAN Clustering

- DBSCAN groups together data points that are close to each other and have a sufficient number of neighbors.
- It is effective in identifying clusters of arbitrary shapes and handling noise in the data.

Scaling of the data:

Scaling

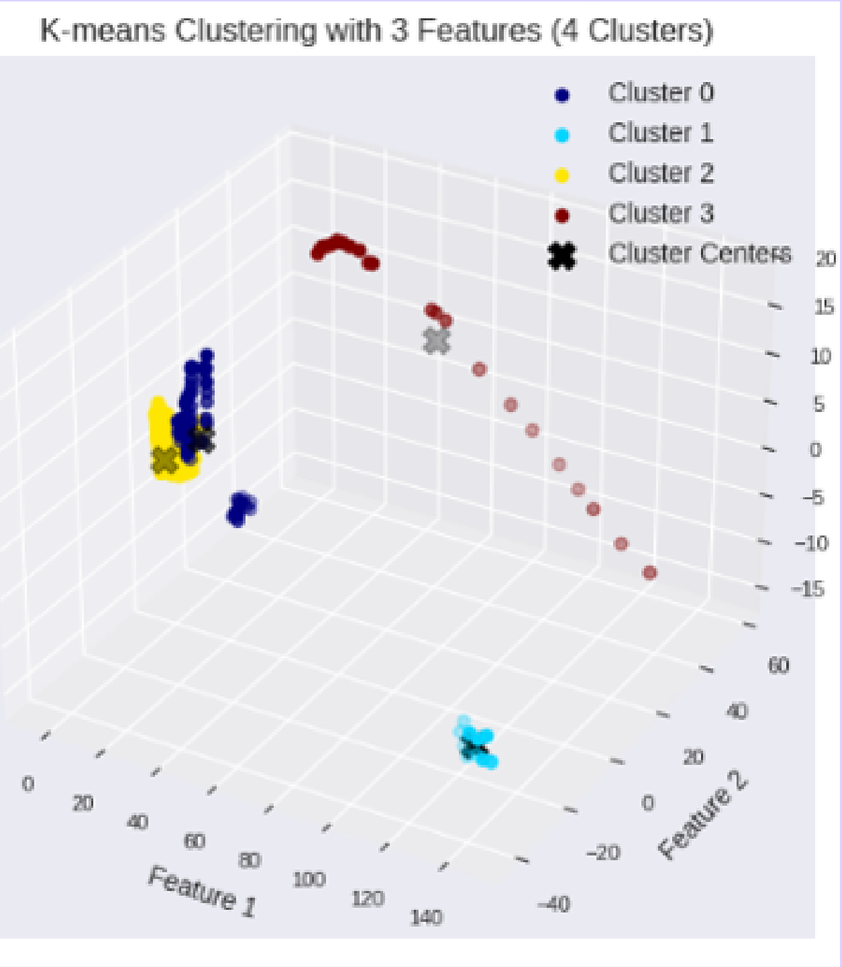
```
In [68]: from sklearn.preprocessing import RobustScaler

In [69]: #Scaling data
         scaled_data = RobustScaler().fit_transform(data_f)

In [70]: scaled_data

Out[70]: array([[-0.05882353,  1.09873672, -0.98837209, ...,  0.11488251,
                 -0.16339439, -0.096184  ],
                [-0.11764706,  1.04585035, -0.98837209, ...,  0.13577023,
                 -0.16392147, -0.09583551],
                [-0.11764706,  1.14122898, -0.98837209, ...,  0.15665796,
                 -0.16102253, -0.0770169 ],
                ...,
                [ 0.        ,  2.53619038,  0.96511628, ...,  0.86684073,
                  0.09882725,  0.74054713],
                [ 0.        ,  2.51480615,  0.96511628, ...,  0.86684073,
                  0.0877586 ,  0.62937794],
                [ 0.        ,  2.75330358,  0.96511628, ...,  0.8694517 ,
                  0.01897483,  0.61648371]])

In [71]: scaled_data.shape

Out[71]: (1743, 15)
```

scaling ensures that the features are on a similar scale, preventing certain features from dominating the learning process and improving the overall performance and stability of machine learning models.

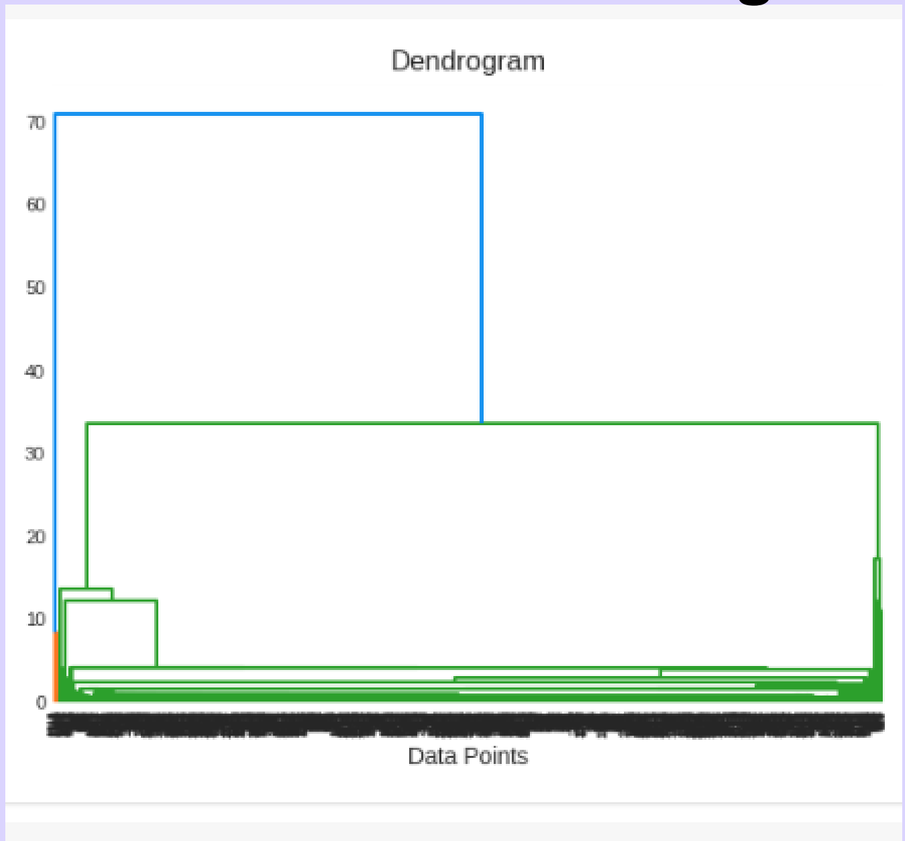
We are proceding to build the model with Robust scaled data as Robust scaling is less affected by extreme values and can provide a more stable scaling in the presence of outliers.

Clustering using PCA data

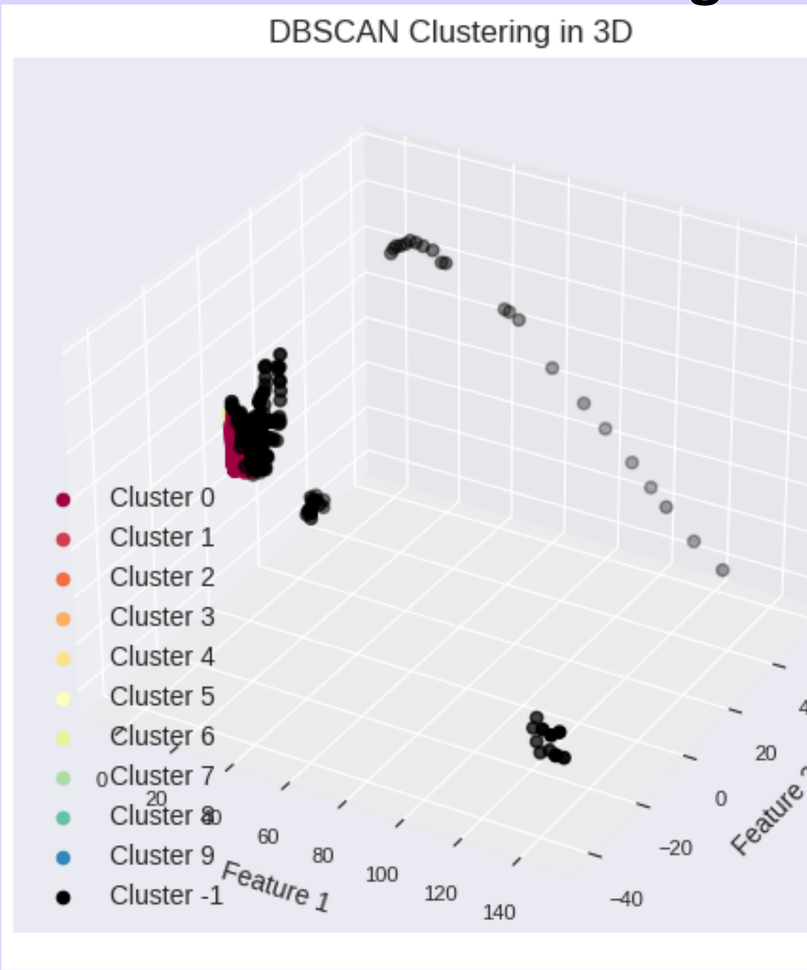
KMeans Clustering



Hierarchical Clustering

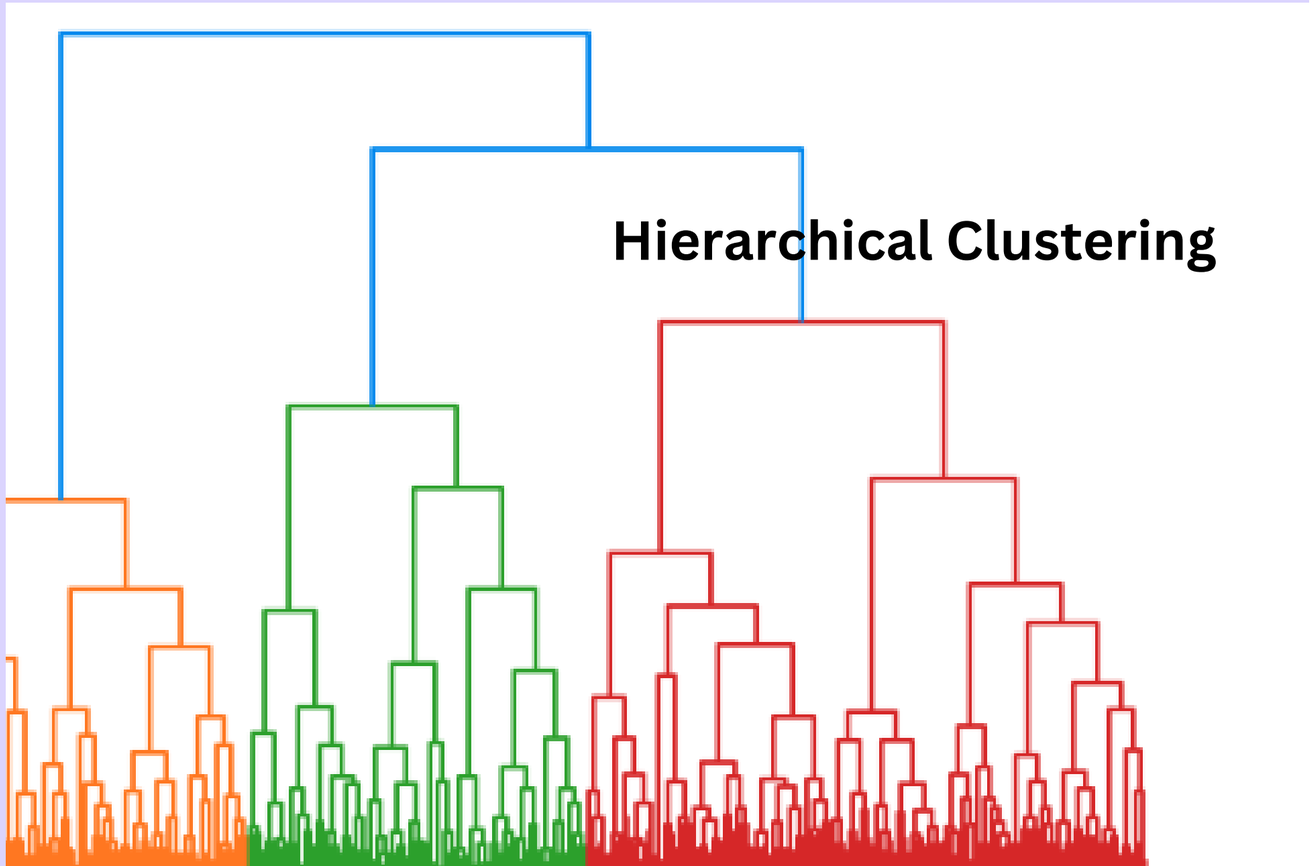


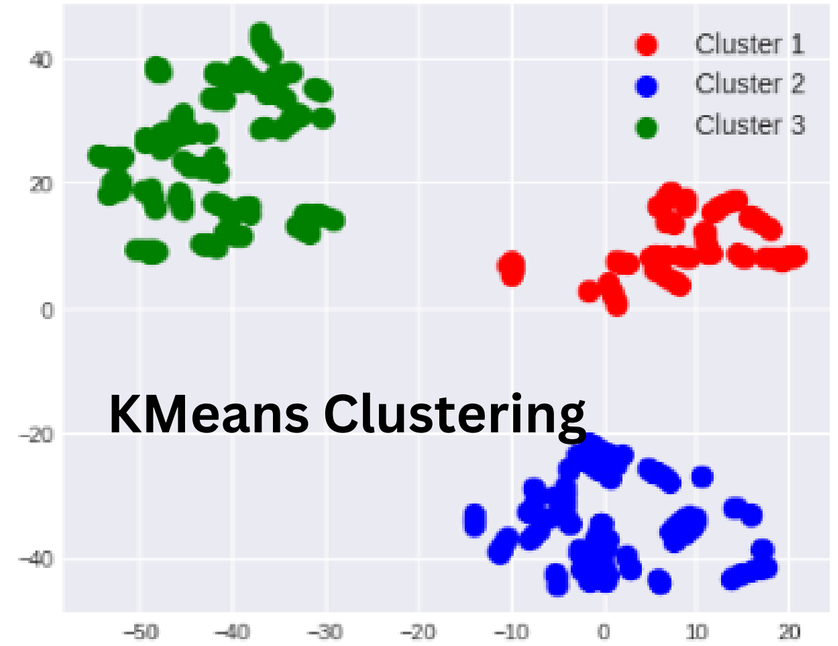
DBSCAN Clustering

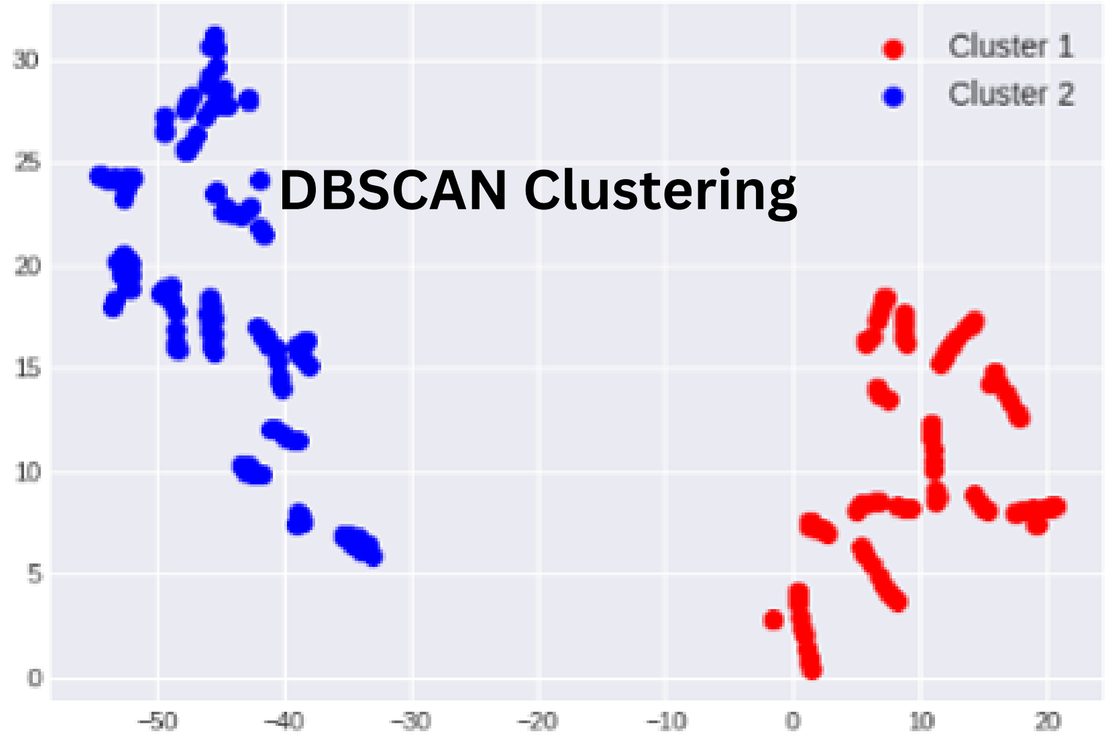


	Model Name	Silhouette Score	Data Distribution
0	PCA		
1	DBSACN Clustering	0.20169	Most are Noise
2	DBSACN Clustering	0.47658	Most are Noise
3	KMeans Clustering(k=3)	0.84369	Unequal
4	KMeans Clustering(k=4)	0.83258	Unequal
5	Hierarchical Clustering	0.94334	Unequal

Clustering using t-SNE data

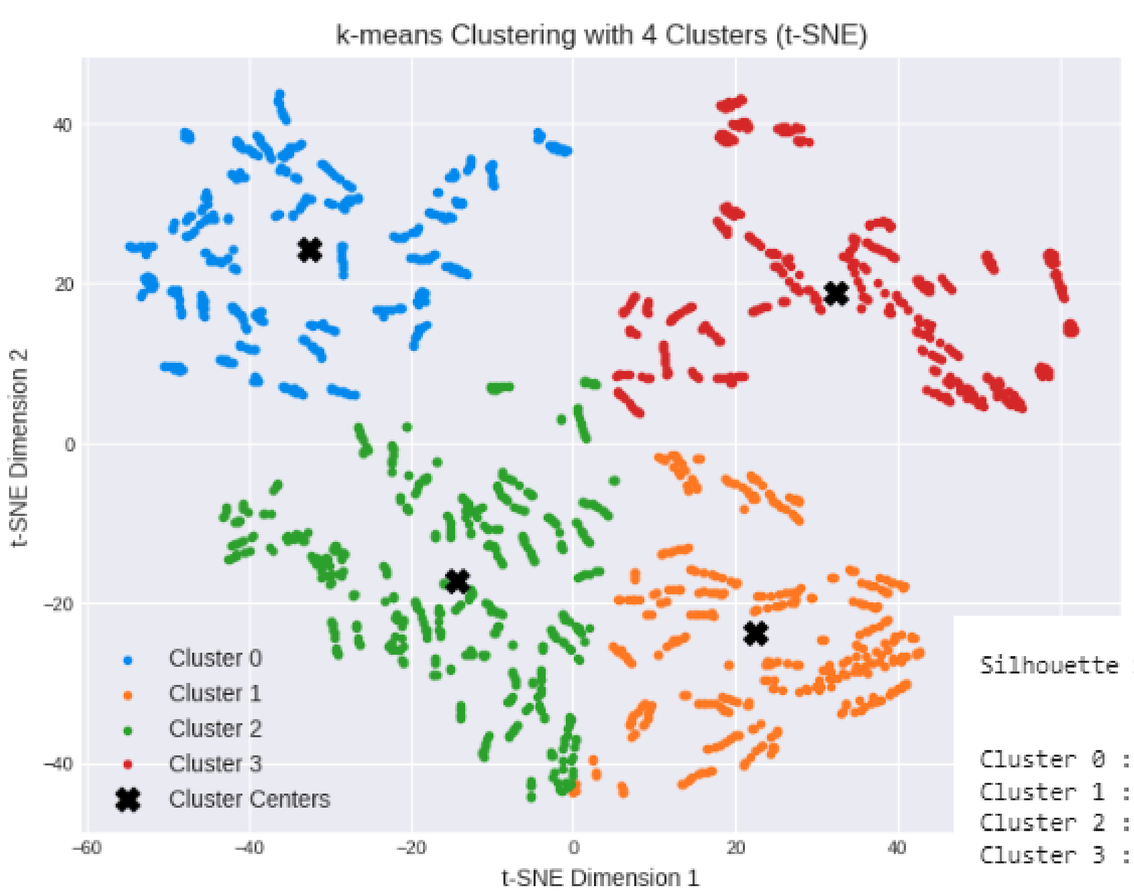






	Model Name	Silhouette Score	Data Distribution
0	PCA		
1	DBSACN Clustering	0.20169	Most are Noise
2	DBSACN Clustering	0.47658	Most are Noise
3	KMeans Clustering(k=3)	0.84369	Unequal
4	KMeans Clustering(k=4)	0.83258	Unequal
5	Hierarchical Clustering	0.94334	Unequal
6	t-SNE		
7	KMeans Clustering(k=3)	0.42291	Equal
8	KMeans Clustering(k=4)	0.45747	Equal
9	KMeans Clustering(k=5)	0.45203	Equal
10	Hierarchy Clustring	0.43433	Equal
11	Hierarchy Clustring	-0.10686	Equal
12	DBSACN Clustering	0.12803	Equal
13	DBSACN Clustering	0.30923	Equal

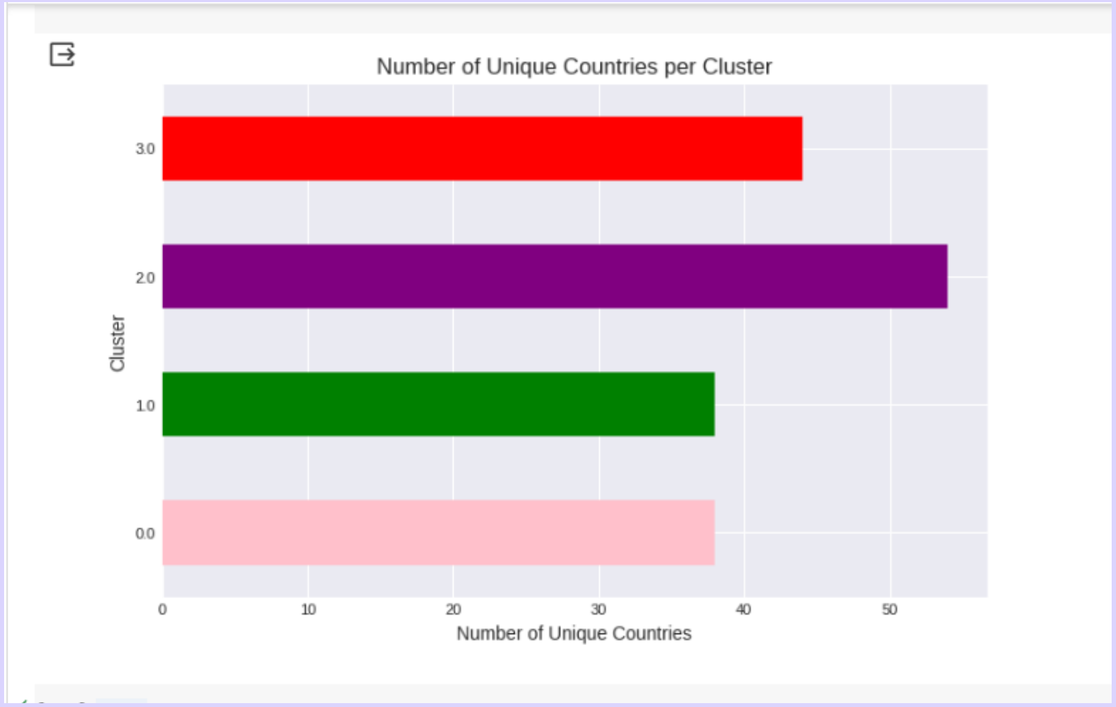
The best Model we got is KMeans Clustering(with k=4) by applying t-SNE(with outliers data)



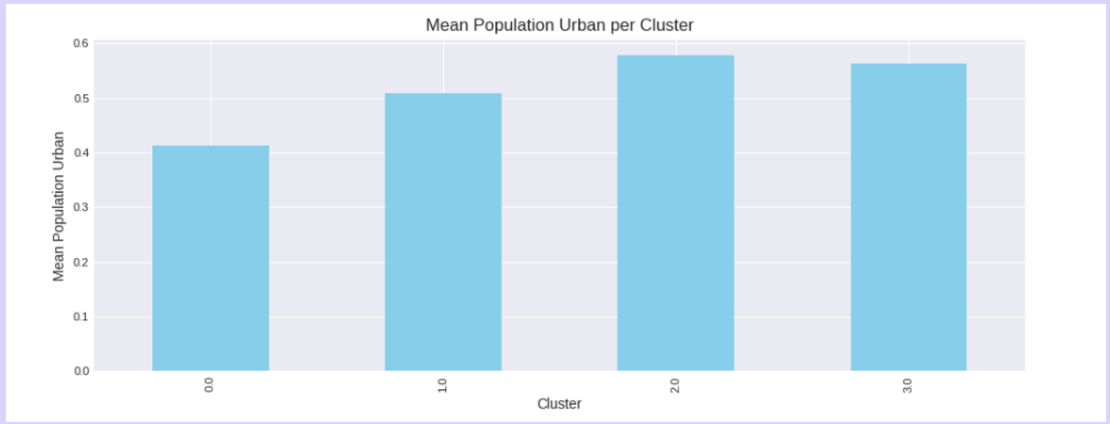

```
Silhouette Score for K-means Clustering: 0.45834067463874817


Cluster 0 : 425 data points
Cluster 1 : 405 data points
Cluster 2 : 499 data points
Cluster 3 : 414 data points
```

Inferences:

```
[179] data_f['Country'].nunique()

      174

[178] print(unique_countries_per_cluster.apply(len))
      print('Total Countries:-',unique_countries_per_cluster.apply(len).sum())

      Cluster
      0.0    38
      1.0    38
      2.0    54
      3.0    44
      Name: Country, dtype: int64
      Total Countries:- 174
```



	Birth Rate	CO2 Emissions	Country	GDP	Health Exp % GDP	Health Exp/Capita	Infant Mortality Rate	Internet Usage	Life Expectancy Female	Life Expectancy Male	Mobile Phone Usage	Population Total	Population Urban	Tourism Inbound	Tourism Outbound
	mean	mean	mean	mean	mean	mean	mean	mean	mean	mean	mean	mean	mean	mean	mean
Cluster															
0.0	0.034039	22290.387900	64.580071	3.177866e+10	0.056851	149.476868	0.061822	0.051601	59.188612	56.056940	0.259431	1.553143e+07	0.412242	1.338851e+09	5.835267e+08
1.0	0.018276	243560.628676	76.867647	3.209072e+11	0.059088	461.694853	0.025875	0.202941	74.095588	68.316176	0.490441	8.926530e+07	0.508919	4.178291e+09	4.414411e+09
2.0	0.019572	228677.365782	101.117994	3.866482e+11	0.075392	1451.324484	0.028581	0.315929	72.681416	68.094395	0.651327	5.689123e+07	0.577867	7.908345e+09	7.494275e+09
3.0	0.021260	142690.542751	95.631970	1.872712e+11	0.065097	1126.973978	0.032156	0.243494	70.449814	65.245353	0.590706	2.756958e+07	0.562093	5.261552e+09	4.476046e+09



Cluster 2

Cluster 1

Cluster 3

Cluster 0

Deployment:

Upload Dataset

Choose a file to upload

Drag and drop file here

Limit 200MB per file • CSV, XLSX

Browse files

Uploaded Data

	Birth Rate	Business Tax Rate	CO2 Emissions	Country	Days to Start Business	Ease of
0	0.02	None	87,931	Algeria	None	
1	0.05	None	9,542	Angola	None	
2	0.043	None	1,617	Benin	None	
3	0.027	None	4,276	Botswana	None	
4	0.046	None	1,041	Burkina Faso	None	
5	0.042	None	301	Burundi	None	
6	0.041	None	3,432	Cameroon	None	
7	0.039	None	268	Central African Republic	None	
8	0.051	None	176	Chad	None	
9	0.039	None	84	Comoros	None	

Correlation Heatmap:

PyplotGlobalUseWarning: You are calling `st.pyplot()` without any arguments. After December 1st,

Select the number of clusters for t-SNE data:

3

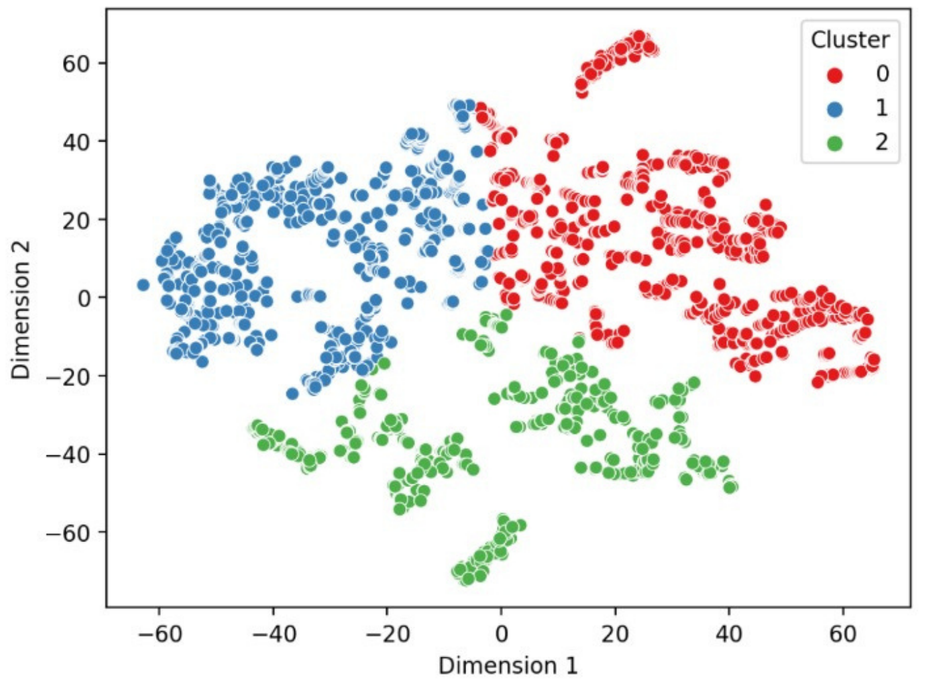
2 10

Select the Clustering Algorithm:

K-Means

Data with Clusters (K-Means)

	Dimension 1	Dimension 2	Cluster
0	1.1786	3.7429	0
1	-50.4739	27.5045	1
2	-47.2089	-3.7824	1
3	-44.7145	3.0944	1
4	-50.8194	-5.2599	1
5	-58.3434	5.5185	1
6	-56.5318	10.5785	1
7	-59.1565	9.4233	1



Silhouette Score: 0.39881065487861633

Thank you!