Unsupervised Learning - Final Project by Marshall Folkman

June 26, 2023

1 Unsupervised Learning - Final Project by Marshall Folkman

1.0.1 Project Description:

The goal of this exercise is to predict the top ten poverty stricken countries that are in greatest need for humanitarian aid. We will do this by performing PCA and building a K-means unsupervised learning model that can be used for future datasets as the world evolves.

Data Source Citation: https://www.kaggle.com/datasets/rohan0301/unsupervised-learning-on-country-data?select=Country-data.csv

Github Source: https://github.com/Vamboozer/AI/tree/main/UnSupervisedML/CountryData-MostPoverty

Video Presentation: https://youtu.be/xVJShH-omXs

1.1 Part 1: Data Cleaning, EDA, and PCA

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.ensemble import IsolationForest

url_CountryData = 'https://raw.githubusercontent.com/Vamboozer/AI/main/
↪UnSupervisedML/CountryData-MostPoverty/Country-data.csv'
url_FeatureDefinitions = 'https://raw.githubusercontent.com/Vamboozer/AI/main/
↪UnSupervisedML/CountryData-MostPoverty/data-dictionary.csv'

CountryData = pd.read_csv(url_CountryData)
FeatureDefinitions = pd.read_csv(url_FeatureDefinitions)
```

[2]:
```python
# display feature definitions
pd.set_option('display.max_colwidth', None)
feature_defs_styled = FeatureDefinitions.style.set_properties(**{'text-align':␣
 ↪'left'})
display(feature_defs_styled)
```

```
<pandas.io.formats.style.Styler at 0x7f1c6e1b1e10>
```

[3]:
```python
# Overview of dataset
print("Shape of Data: ", CountryData.shape)
display(CountryData.head(12))
```

```
Shape of Data:  (167, 10)
```

	country	child_mort	exports	health	imports	income
0	Afghanistan	90.2	10.0	7.58	44.9	1610
1	Albania	16.6	28.0	6.55	48.6	9930
2	Algeria	27.3	38.4	4.17	31.4	12900
3	Angola	119.0	62.3	2.85	42.9	5900
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100
5	Argentina	14.5	18.9	8.10	16.0	18700
6	Armenia	18.1	20.8	4.40	45.3	6700
7	Australia	4.8	19.8	8.73	20.9	41400
8	Austria	4.3	51.3	11.00	47.8	43200
9	Azerbaijan	39.2	54.3	5.88	20.7	16000
10	Bahamas	13.8	35.0	7.89	43.7	22900
11	Bahrain	8.6	69.5	4.97	50.9	41100

	inflation	life_expec	total_fer	gdpp
0	9.440	56.2	5.82	553
1	4.490	76.3	1.65	4090
2	16.100	76.5	2.89	4460
3	22.400	60.1	6.16	3530
4	1.440	76.8	2.13	12200
5	20.900	75.8	2.37	10300
6	7.770	73.3	1.69	3220
7	1.160	82.0	1.93	51900
8	0.873	80.5	1.44	46900
9	13.800	69.1	1.92	5840
10	-0.393	73.8	1.86	28000
11	7.440	76.0	2.16	20700

1.1.1 EDA

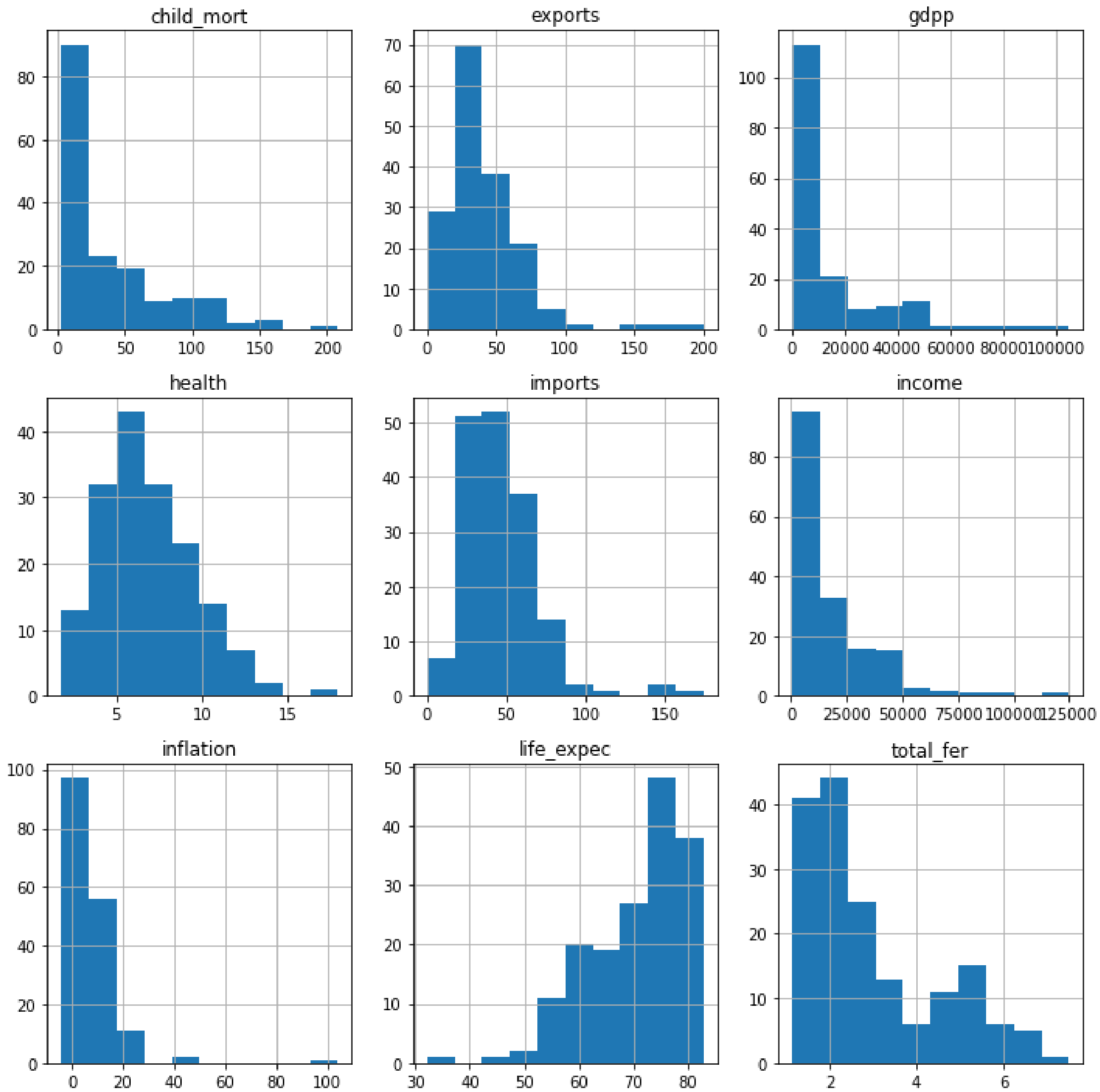
[4]:
```
# Summary statistics
display(CountryData.describe())
```

```
        child_mort     exports      health     imports         income  \
count  167.000000  167.000000  167.000000  167.000000     167.000000
mean    38.270060   41.108976    6.815689   46.890215   17144.688623
std     40.328931   27.412010    2.746837   24.209589   19278.067698
min      2.600000    0.109000    1.810000    0.065900     609.000000
25%      8.250000   23.800000    4.920000   30.200000    3355.000000
50%     19.300000   35.000000    6.320000   43.300000    9960.000000
75%     62.100000   51.350000    8.600000   58.750000   22800.000000
max    208.000000  200.000000   17.900000  174.000000  125000.000000

        inflation  life_expec   total_fer           gdpp
count  167.000000  167.000000  167.000000     167.000000
mean     7.781832   70.555689    2.947964   12964.155689
std     10.570704    8.893172    1.513848   18328.704809
min     -4.210000   32.100000    1.150000     231.000000
25%      1.810000   65.300000    1.795000    1330.000000
50%      5.390000   73.100000    2.410000    4660.000000
75%     10.750000   76.800000    3.880000   14050.000000
max    104.000000   82.800000    7.490000  105000.000000
```

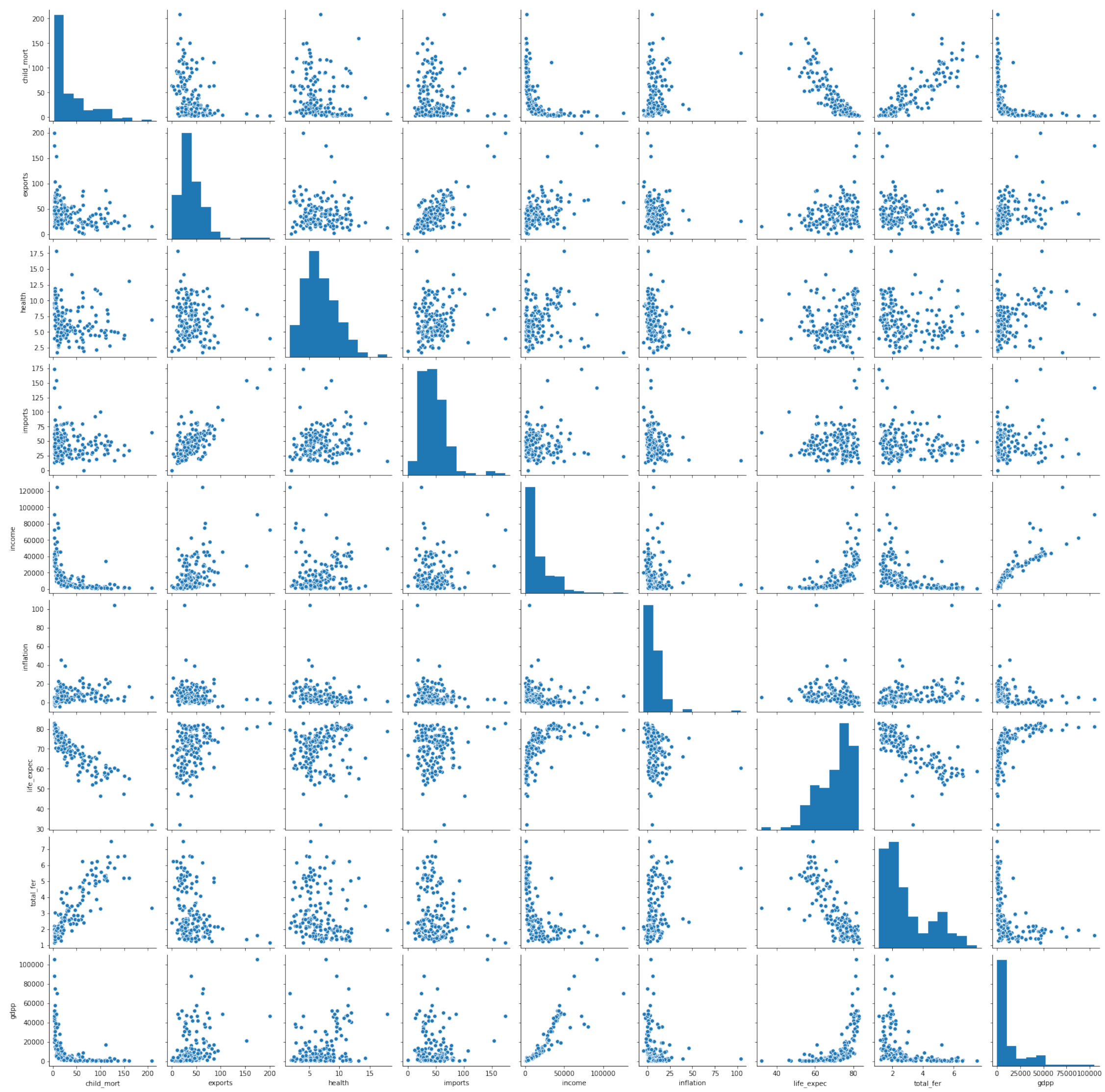
[5]:
```
# Checking for missing values
print(CountryData.isnull().sum())
```

```
country       0
child_mort    0
exports       0
health        0
imports       0
income        0
inflation     0
life_expec    0
total_fer     0
gdpp          0
dtype: int64
```

[6]:
```
# Histograms to understand distributions
CountryData.hist(figsize=(10,10))
plt.tight_layout()
plt.show()
```



[7]:
```python
# Pairplot to visualize correlations between variables
sns.pairplot(CountryData.drop("country", axis=1))  # we drop 'country' because it is a non-numeric column
plt.show()
```



[8]:
```
# Standardize the data (excluding 'country')
scaler = StandardScaler()
CountryData_scaled = scaler.fit_transform(CountryData.drop('country', axis=1))

# Apply PCA
pca = PCA()
CountryData_pca = pca.fit_transform(CountryData_scaled)

# Print the explained variance ratio
display("Explained variance ratio: ", pca.explained_variance_ratio_)

# Cumulative explained variance
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
```

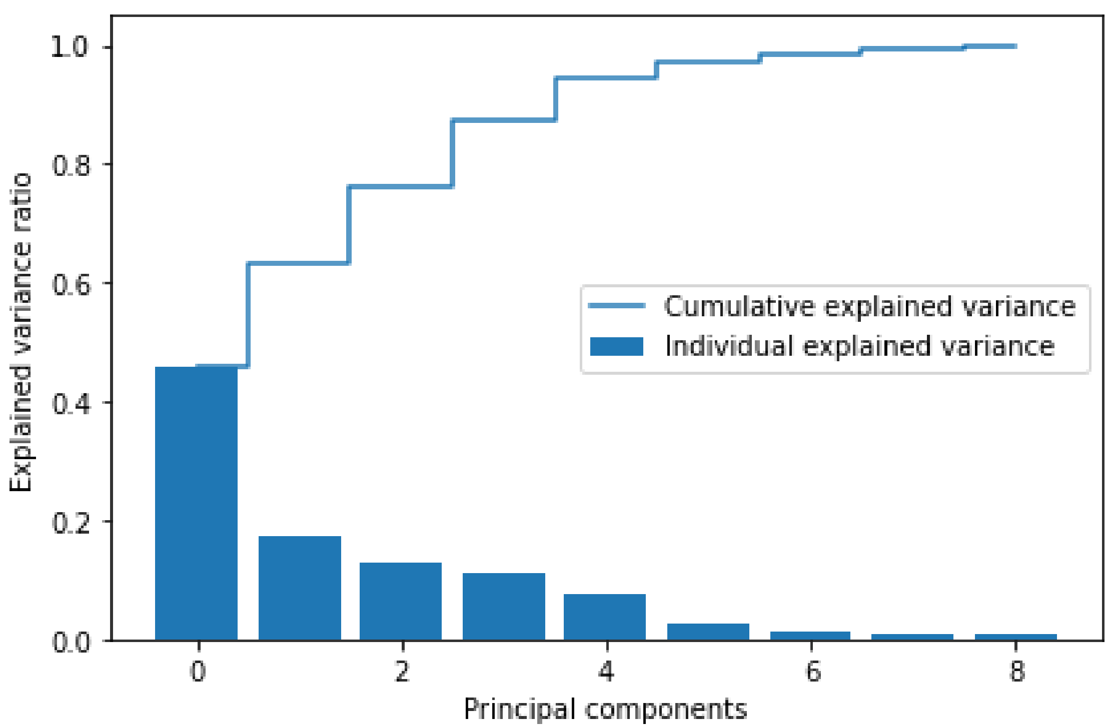
```python
display("Cumulative explained variance: ", cumulative_variance)

# Plot the explained variance
plt.figure(figsize=(6,4))
plt.bar(range(len(pca.explained_variance_ratio_)), pca.
 ↪explained_variance_ratio_, align='center', label='Individual explained␣
 ↪variance')
plt.step(range(len(cumulative_variance)), cumulative_variance,␣
 ↪where='mid',label='Cumulative explained variance')
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal components')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```

```
'Explained variance ratio: '
```

```
array([0.4595174 , 0.17181626, 0.13004259, 0.11053162, 0.07340211,
       0.02484235, 0.0126043 , 0.00981282, 0.00743056])
```

```
'Cumulative explained variance: '
```

```
array([0.4595174 , 0.63133365, 0.76137624, 0.87190786, 0.94530998,
       0.97015232, 0.98275663, 0.99256944, 1.        ])
```

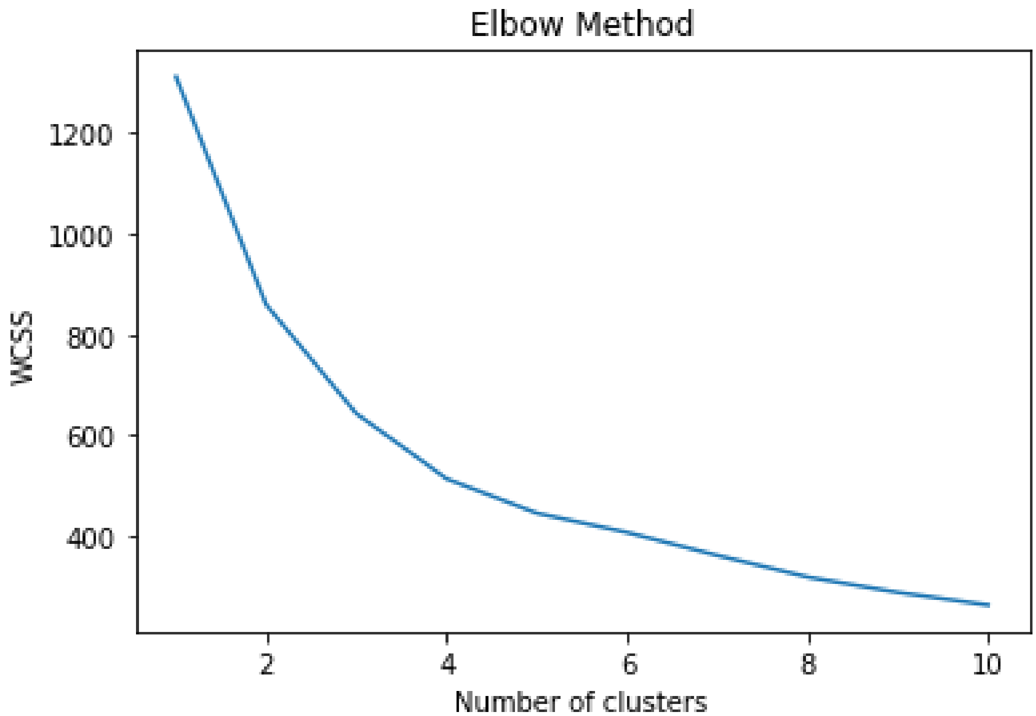


We can see here that if we include the first four components we're explaining approximately 87% of the total variance with only these four components. 87% should be sufficient here so lets go with that to keep our model lean and simple.

1.1.2 Select, Build, and Train Model

Lets try out a K-means clustering model because it is simple, flexible, and should work well for only four principle components selected in the previous step.

[9]:
```
# Select the first four principal components
CountryData_pca_4 = CountryData_pca[:,:4]

# Determine the number of clusters
# A common technique is the elbow method, where the sum of squared distances to␣
 ↪the nearest cluster center
# (within-cluster sum of squares, or WCSS) is calculated for different numbers␣
 ↪of clusters, and the number of
# clusters where adding another cluster doesn't significantly improve WCSS is␣
 ↪selected.

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(CountryData_pca_4)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# From the plot, we choose the number of clusters where the decrease in WCSS␣
 ↪starts to level off (the "elbow")
```



[10]:

```python
# Suppose the optimal number of clusters is 3
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)
clusters = kmeans.fit_predict(CountryData_pca_4)

# Add the cluster assignments to the original dataframe
print(CountryData.shape)
CountryData['Cluster'] = clusters
print(CountryData.shape)

# Calculate mean values for each cluster
cluster_means = CountryData.groupby('Cluster').mean()
display(cluster_means)
```

```
(167, 10)
(167, 11)

         child_mort    exports    health    imports        income  inflation  \
Cluster
0         91.610417  29.571042  6.433542  43.133333   3897.354167  11.911146
1          4.897143  58.431429  8.917429  51.508571  45802.857143   2.535000
2         21.695238  40.484393  6.158333  47.112689  12773.690476   7.608405

         life_expec  total_fer          gdpp
Cluster
```

```
0          59.239583   4.992083   1909.208333
1          80.245714   1.741143  43117.142857
2          72.984524   2.282738   6717.523810
```

The mean values above help us interpret the clusters. For example, a cluster with high child mortality and low income and GDP could be considered as a group of countries in severe poverty and in need of aid. In contrast, a cluster with low child mortality and high income and GDP could be considered as a group of wealthy, developed countries.

[11]:
```python
# Get countries belonging to cluster 0
poverty_cluster = CountryData[CountryData['Cluster'] == 0]

# Sort countries in the poverty cluster by child mortality, in descending order
poverty_cluster_sorted = poverty_cluster.sort_values(by='child_mort',␣
↪ascending=False)

# Display the sorted countries
print("Number of countries in the poverty cluster: ", poverty_cluster_sorted.
↪shape[0], "\n")
print(poverty_cluster_sorted['country'])
```

```
Number of countries in the poverty cluster:  48

66                        Haiti
132                Sierra Leone
32                         Chad
31     Central African Republic
97                         Mali
113                     Nigeria
112                       Niger
3                        Angola
25                 Burkina Faso
37             Congo, Dem. Rep.
64                Guinea-Bissau
40                Cote d'Ivoire
17                        Benin
49            Equatorial Guinea
63                       Guinea
28                     Cameroon
106                  Mozambique
87                      Lesotho
99                   Mauritania
26                      Burundi
116                    Pakistan
94                       Malawi
150                        Togo
0                   Afghanistan
```

```
88                        Liberia
36                        Comoros
166                        Zambia
155                        Uganda
56                         Gambia
84                            Lao
142                         Sudan
59                          Ghana
147                      Tanzania
129                       Senegal
38                    Congo, Rep.
55                          Gabon
126                        Rwanda
81                       Kiribati
149                   Timor-Leste
80                          Kenya
93                     Madagascar
165                         Yemen
108                       Namibia
50                        Eritrea
137                  South Africa
21                       Botswana
72                           Iraq
136               Solomon Islands
Name: country, dtype: object
```

[12]:

```
display(cluster_means)
```

```
         child_mort    exports    health    imports        income  inflation  \
Cluster
0         91.610417  29.571042  6.433542  43.133333   3897.354167  11.911146
1          4.897143  58.431429  8.917429  51.508571  45802.857143   2.535000
2         21.695238  40.484393  6.158333  47.112689  12773.690476   7.608405

         life_expec  total_fer          gdpp
Cluster
0         59.239583   4.992083   1909.208333
1         80.245714   1.741143  43117.142857
2         72.984524   2.282738   6717.523810
```

[13]:

```
# Select only the countries from the "needs aid" cluster in the PCA transformed␣
↪space
aid_cluster_countries_pca = CountryData_pca_4[CountryData['Cluster'] == 0]

# Calculate the Euclidean distance from each country to the centroid of the␣
↪"needs aid" cluster
aid_cluster_center = kmeans.cluster_centers_[0]
```

```python
distances = np.sqrt(((aid_cluster_countries_pca - aid_cluster_center)**2).sum(axis=1))

# Convert distances to a pandas series
distances_series = pd.Series(distances, index=CountryData[CountryData['Cluster'] == 0].index)

# Select the 10 countries with the smallest distances to the centroid
closest_countries = distances_series.nsmallest(10).index

# Display these countries with their corresponding original data
closest_countries_data = CountryData.loc[closest_countries]
display(closest_countries_data)
```

	country	child_mort	exports	health	imports	income	inflation
17	Benin	111.0	23.8	4.10	37.2	1820	0.885
56	Gambia	80.3	23.8	5.69	42.7	1660	4.300
166	Zambia	83.1	37.0	5.89	30.9	3280	14.000
28	Cameroon	108.0	22.2	5.13	27.0	2660	1.910
94	Malawi	90.5	22.8	6.59	34.9	1030	12.100
147	Tanzania	71.9	18.7	6.01	29.1	2090	9.250
106	Mozambique	101.0	31.5	5.21	46.2	918	7.640
40	Cote d'Ivoire	111.0	50.6	5.30	43.3	2690	5.390
63	Guinea	109.0	30.3	4.93	43.2	1190	16.100
59	Ghana	74.7	29.5	5.22	45.9	3060	16.600

	life_expec	total_fer	gdpp	Cluster
17	61.8	5.36	758	0
56	65.5	5.71	562	0
166	52.0	5.40	1460	0
28	57.3	5.11	1310	0
94	53.1	5.31	459	0
147	59.3	5.43	702	0
106	54.5	5.56	419	0
40	56.3	5.27	1220	0
63	58.0	5.34	648	0
59	62.2	4.27	1310	0

1.1.3 Build Model for future datasets

Great! This model seems to be appropriate for the situation. Now lets better construct this model to be used again in the future as these facts about these countries may change from time to time. This is especially true if the countries in most need of aid are the countries always getting the aid.

We basically need to repeat most everything we just did, but this time we need it automated. We need an algorithm that automatically identifies the cluster that corresponds with impoverished countries. Based on the very strong correlation with impoverished countries and low income per person, I believe a very robust strategy would be to simply select the cluster with the lowest mean income per person.

[14]:
```python
def cluster_and_identify_countries(df):
    # Standardize the data (excluding 'country')
    scaler = StandardScaler()
    df_normalized = scaler.fit_transform(df.drop('country', axis=1))

    # Apply PCA
    pca = PCA()
    df_pca = pca.fit_transform(df_normalized)

    # Select the first four principal components
    df_pca_4 = df_pca[:,:4]

    # Already determined that the optimal number of clusters is 3
    kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)
    clusters = kmeans.fit_predict(df_pca_4)

    # Add the cluster assignments to the original dataframe
    df['Cluster'] = clusters

    # Calculate mean values for each cluster
    cluster_means = df.groupby('Cluster').mean()
    display(cluster_means)

    # Identify the "needs aid" cluster (the one with the lowest mean income)
    aid_cluster_id = df.groupby('Cluster')['income'].mean().idxmin()
    print("The cluster that corresponds to impoverished countries is cluster",␣
↪aid_cluster_id)

    # Filter the dataframe to include only countries in the "needs aid" cluster
    aid_cluster_countries = df[df['Cluster'] == aid_cluster_id]

    # Calculate the Euclidean distance from each country to the centroid of the␣
↪"needs aid" cluster
    aid_cluster_center = kmeans.cluster_centers_[aid_cluster_id]
    distances = np.sqrt(((df_pca_4[df['Cluster'] == aid_cluster_id] -␣
↪aid_cluster_center)**2).sum(axis=1))

    # Add the distances to the dataframe
    aid_cluster_countries = aid_cluster_countries.assign(Distance=distances)

    # Select the 10 countries with the smallest distances to the centroid
    top_countries = aid_cluster_countries.nsmallest(10, 'Distance')
```

```python
    return top_countries
```

[15]:
```python
# Test the new automated model with the same dataset (we expect the same
↪results)
new_data = pd.read_csv(url_CountryData)
top_countries = cluster_and_identify_countries(new_data)
display(top_countries)
```

```
         child_mort    exports    health    imports        income  inflation  \
Cluster
0         91.610417  29.571042  6.433542  43.133333   3897.354167  11.911146
1          4.897143  58.431429  8.917429  51.508571  45802.857143   2.535000
2         21.695238  40.484393  6.158333  47.112689  12773.690476   7.608405

         life_expec  total_fer          gdpp
Cluster
0         59.239583   4.992083   1909.208333
1         80.245714   1.741143  43117.142857
2         72.984524   2.282738   6717.523810


The cluster that corresponds to impoverished countries is cluster 0

           country  child_mort  exports  health  imports  income  inflation  \
17           Benin       111.0     23.8    4.10     37.2    1820      0.885
56          Gambia        80.3     23.8    5.69     42.7    1660      4.300
166         Zambia        83.1     37.0    5.89     30.9    3280     14.000
28        Cameroon       108.0     22.2    5.13     27.0    2660      1.910
94          Malawi        90.5     22.8    6.59     34.9    1030     12.100
147       Tanzania        71.9     18.7    6.01     29.1    2090      9.250
106     Mozambique       101.0     31.5    5.21     46.2     918      7.640
40   Cote d'Ivoire       111.0     50.6    5.30     43.3    2690      5.390
63          Guinea       109.0     30.3    4.93     43.2    1190     16.100
59           Ghana        74.7     29.5    5.22     45.9    3060     16.600

     life_expec  total_fer  gdpp  Cluster  Distance
17         61.8       5.36   758        0  0.501046
56         65.5       5.71   562        0  0.549240
166        52.0       5.40  1460        0  0.595864
28         57.3       5.11  1310        0  0.597193
94         53.1       5.31   459        0  0.643892
147        59.3       5.43   702        0  0.697322
106        54.5       5.56   419        0  0.711408
40         56.3       5.27  1220        0  0.806861
63         58.0       5.34   648        0  0.820855
59         62.2       4.27  1310        0  0.869650
```

1.2 Conclusion

In this problem, we utilized Principal Component Analysis (PCA) and K-means clustering to identify countries most in need of aid based on various socio-economic indicators. PCA was instrumental in reducing the dimensionality of our dataset while preserving its essential structure and variations. This allowed us to simplify the dataset, reducing computational complexity, and removing redundant information. The K-means algorithm helped us cluster the countries into different groups based on their socio-economic characteristics. It was then straightforward to identify the cluster of countries with the lowest mean income as the most impoverished and, therefore, the most in need of aid. The distance calculation further provided a means to rank countries within this impoverished cluster according to their closeness to the cluster centroid, i.e., their representative socio-economic condition. The resulting model provided an automated, data-driven approach to aid prioritization, demonstrating the power and utility of unsupervised learning techniques in informing policy decisions. From this analysis, countries like Benin, Gambia, and Zambia emerged as top candidates for aid, highlighting the disparities in global socio-economic conditions and the need for targeted interventions.