Final Project for Course: Python Foundations

Project Topic: Using Unsupervised Machine Learning to Cluster Countries According to Their Needs

Submitted By:

Gibson Opurum (202201062)

Anthony Ogon (202202072)

Instructor: Atlas Khan

Date: Friday, February 4, 2022

In partial fulfilment of the MSc in Computer Science, Machine Learning, and Artificial Intelligence Program

1.0 Abstract

Classify countries based on socioeconomic and health factors that influence the country's overall development.

The dataset for this analysis was collected by a non-profit called Help International. They want to find a solution to assist countries below the global poverty line.

We will limit our analysis to using KMeans clustering and the Hierarchical clustering algorithm to find countries that need the most help.

1.1 Introduction

HELP International is an *international* humanitarian organization dedicated to eradicating poverty and providing basic amenities and aid to individuals in developing nations during disasters and natural disasters.

Our analysis will help, HELP International raise approximately $10 million dollars to alleviate poverty in the world's developing countries. But the NGO's CEO needs to figure out how to spend this money wisely and strategically.

The CEO must decide which countries are most desperate for assistance. As a result, our job as data scientists is to categorize countries based on socioeconomic and health characteristics that influence the country's overall development. Then we could advise which nations the CEO should concentrate on the most.

2.0 Methodology

```
#Load ML Python libraries

import pandas as pd
import numpy as np
import io
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as exp
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans,AgglomerativeClustering
from sklearn.metrics import silhouette_score
```

```
# chose the csv files from local directory (csv with raw data downloaded in email
# (click on the "choose files" button after running the cell)
from google.colab import files
uploaded = files.upload()
```

Choose Files 2 files

- **Country-data.csv**(application/vnd.ms-excel) - 9229 bytes, last modified: 12/11/2021 - 100% done
- **data-dictionary.csv**(application/vnd.ms-excel) - 808 bytes, last modified: 12/11/2021 - 100% done

```
Saving Country-data.csv to Country-data.csv
Saving data-dictionary.csv to data-dictionary.csv
```

```
#The the first csv files
df = pd.read_csv(
    io.BytesIO(uploaded['Country-data.csv']), encoding='utf-8')
df
```

	country	child_mort	exports	health	imports	income	inflation	life_e
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	
...	...	...	...	...	...	...	...	
162	Vanuatu	29.2	46.6	5.25	52.7	2950	2.62	
163	Venezuela	17.1	28.5	4.91	17.6	16500	45.90	
164	Vietnam	23.3	72.0	6.84	80.2	4490	12.10	
165	Yemen	56.3	30.0	5.18	34.4	4480	23.60	
166	Zambia	83.1	37.0	5.89	30.9	3280	14.00	

167 rows × 10 columns

```
#Get the second csv file. This describes the columns in the dataset.
data_dict = pd.read_csv(
    io.BytesIO(uploaded['data-dictionary.csv']), encoding='utf-8')
data_dict
```

	Column Name	Description
0	country	Name of the country
1	child_mort	Death of children under 5 years of age per 100...
2	exports	Exports of goods and services per capita. Give...
3	health	Total health spending per capita. Given as %ag...

```
#Check for missing values
print(df.shape)
print(df.isna().sum())
```

```
(167, 10)
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
```

```
#Drop duplicates rows from dataframe
df.drop_duplicates(inplace=True)

print(df.shape)
```

```
(167, 10)
```

3.0 Exploratory Data Analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. Source: [Towards Data Science.]

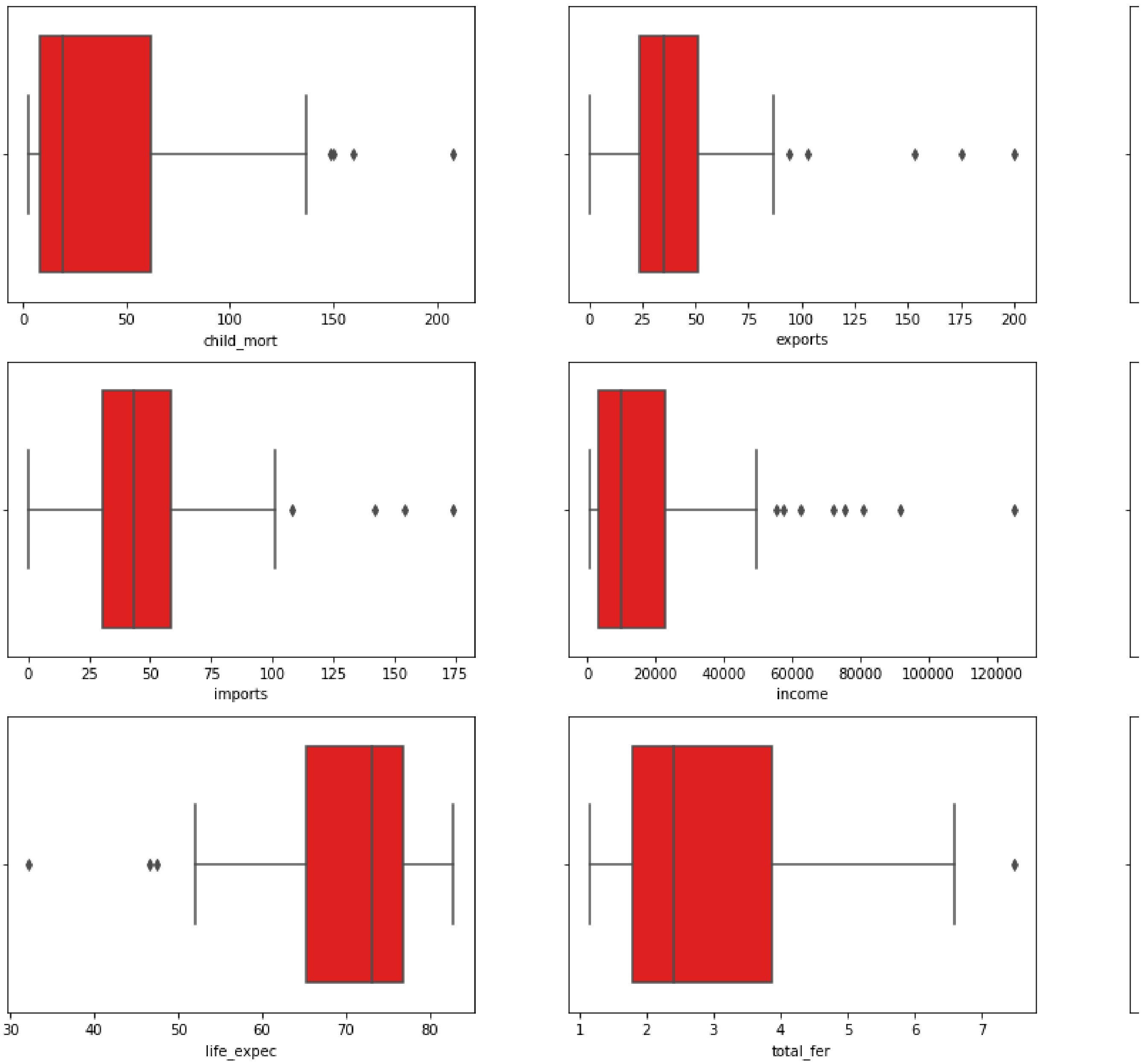
```
df.columns
```

```
Index(['country', 'child_mort', 'exports', 'health', 'imports', 'income',
       'inflation', 'life_expec', 'total_fer', 'gdpp'],
      dtype='object')
```

3.1 Univariate Analysis

Look for outliers in data columns.

```
fig,axs=plt.subplots(3,3,figsize=(20,13))
col=['child_mort', 'exports', 'health', 'imports', 'income','inflation', 'life_exp
ax=axs.flatten()
for i,j in enumerate(col):
    sns.boxplot(x=df[j],ax=ax[i],color='red')
```



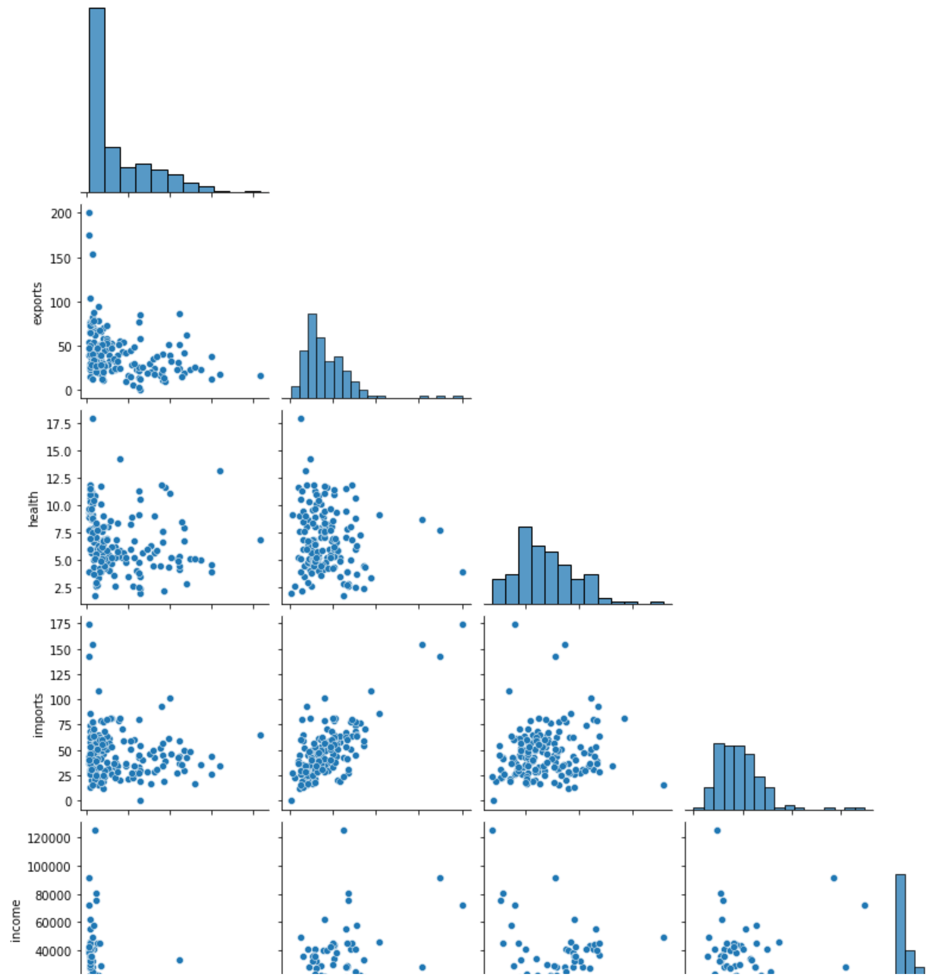
The graph above depicts our data distribution and the numerous outliers for each feature set. It's worth noting that not all of the features are dispersed evenly. The majority of the data, on the other hand, is normally distributed.

3.2 Multivariate Analysis

Let's examine the relationship between two or more variables and identifies which, if any, are related to a certain outcome.

```
sns.pairplot(df,corner=True)
```

```
<seaborn.axisgrid.PairGrid at 0x7fdc1ff3d2d0>
```



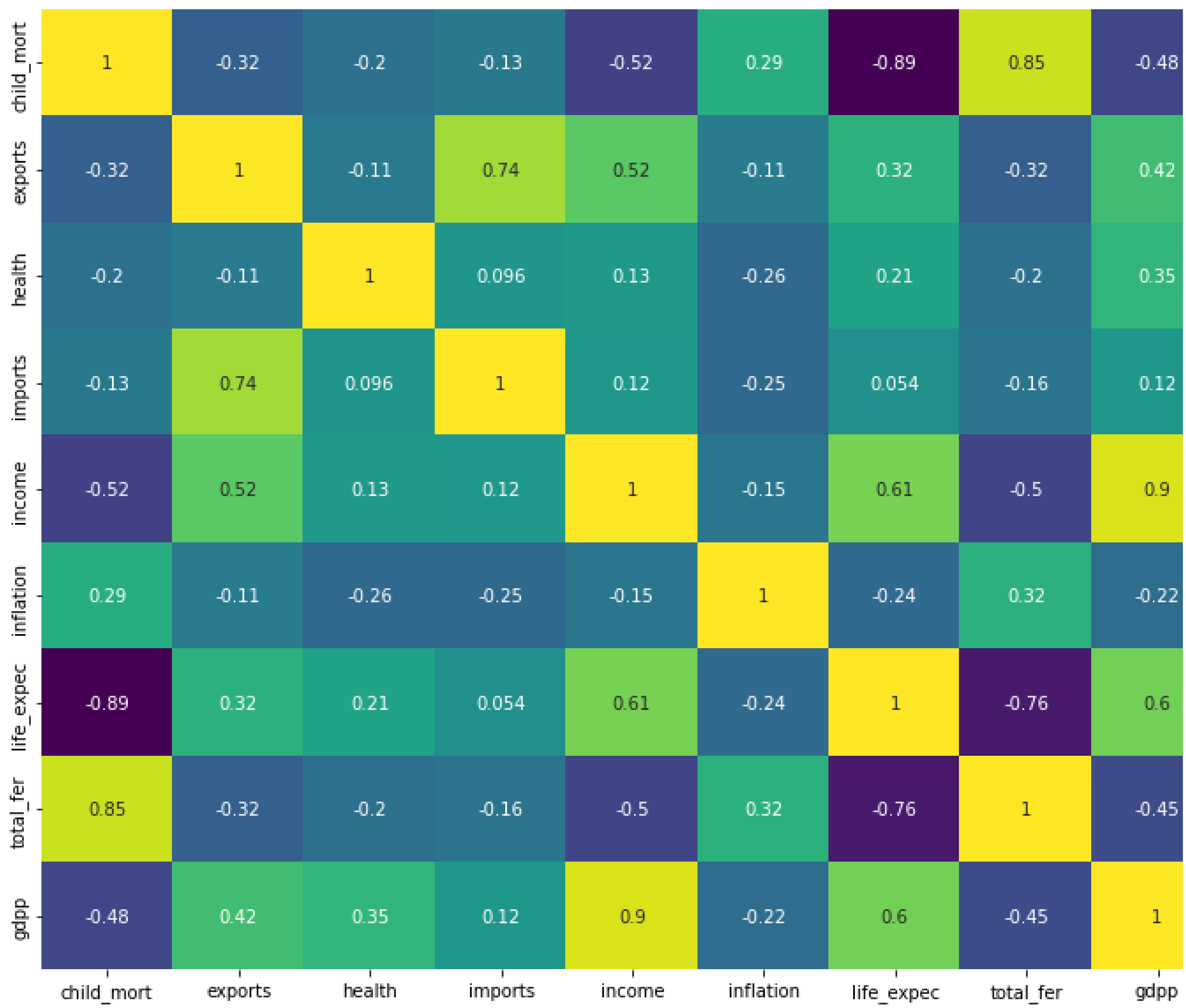
As shown in the graph above, there appear to be several relationships. For instance, total_fer is strongly correlated with child_mort while life_expec is negatively correlated with child_mort. But life_expec seem to have a weak correlation with exports. We can say the same thing for most other features of the pairwise plot.

4.0 Correlation

```
#Get data set correlation statistics
df.corr()
```

	child_mort	exports	health	imports	income	inflation	life_expec
child_mort	1.000000	-0.318093	-0.200402	-0.127211	-0.524315	0.288276	-0.886676
exports	-0.318093	1.000000	-0.114408	0.737381	0.516784	-0.107294	0.316313
health	-0.200402	-0.114408	1.000000	0.095717	0.129579	-0.255376	0.210692
imports	-0.127211	0.737381	0.095717	1.000000	0.122406	-0.246994	0.054391

```
#Use a heatmap to visualize the dataset correlation
fig = plt.figure(figsize=(15,10))
ax = sns.heatmap(df.corr(),annot=True,cmap = 'viridis')
plt.show()
```



We can see that the different colors show different correlations for our data from the heatmap. For example, yellow means strong correlation, whereas purple shows weak correlation.

5.0 Data Preprocessing

```
df.head(5)
```

	country	child_mort	exports	health	imports	income	inflation	life_expe
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76

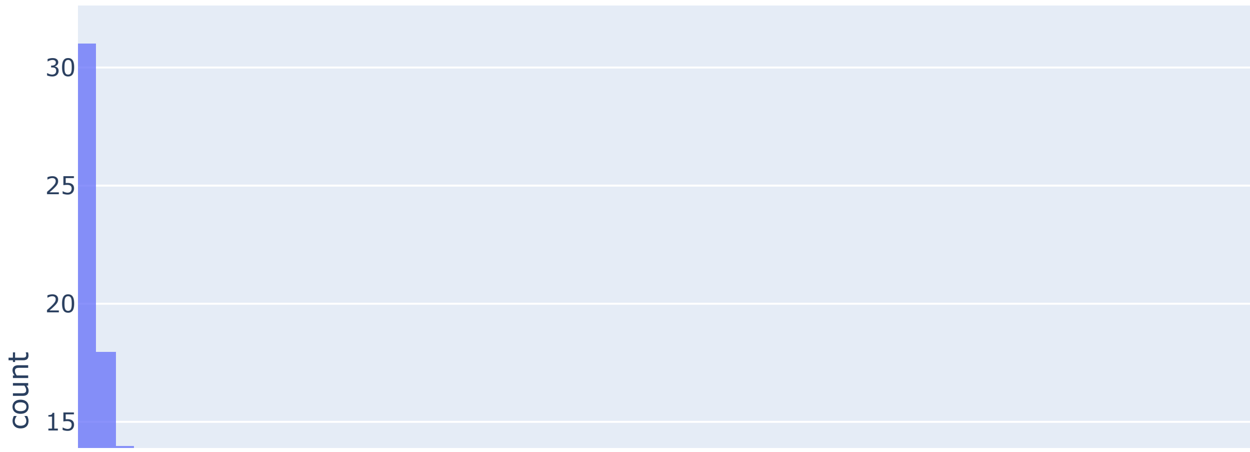
```
#Scale the data set excluding the country column.
data=df.drop('country',axis=1)
```

```
# #Import min-max scaler for scaling the dataframe.
# from sklearn.preprocessing import MinMaxScaler
# scalar = MinMaxScaler()

#Use standard scaler instead.
from sklearn.preprocessing import StandardScaler
s=StandardScaler()
data=s.fit_transform(data);
```

6.0 Data Visualization

```
#Get the GDP count
exp.histogram(data_frame=df,x = 'gdpp',nbins=167,opacity=0.75,barmode='overlay')
```



From the above plot, we see that the nations whose gross domestic product is less than 10k are many; up to 30 countries fall within this bracket, whereas fewer countries have GDP above 100k. Remember, our goal is to find out the developing countries that need help the most.

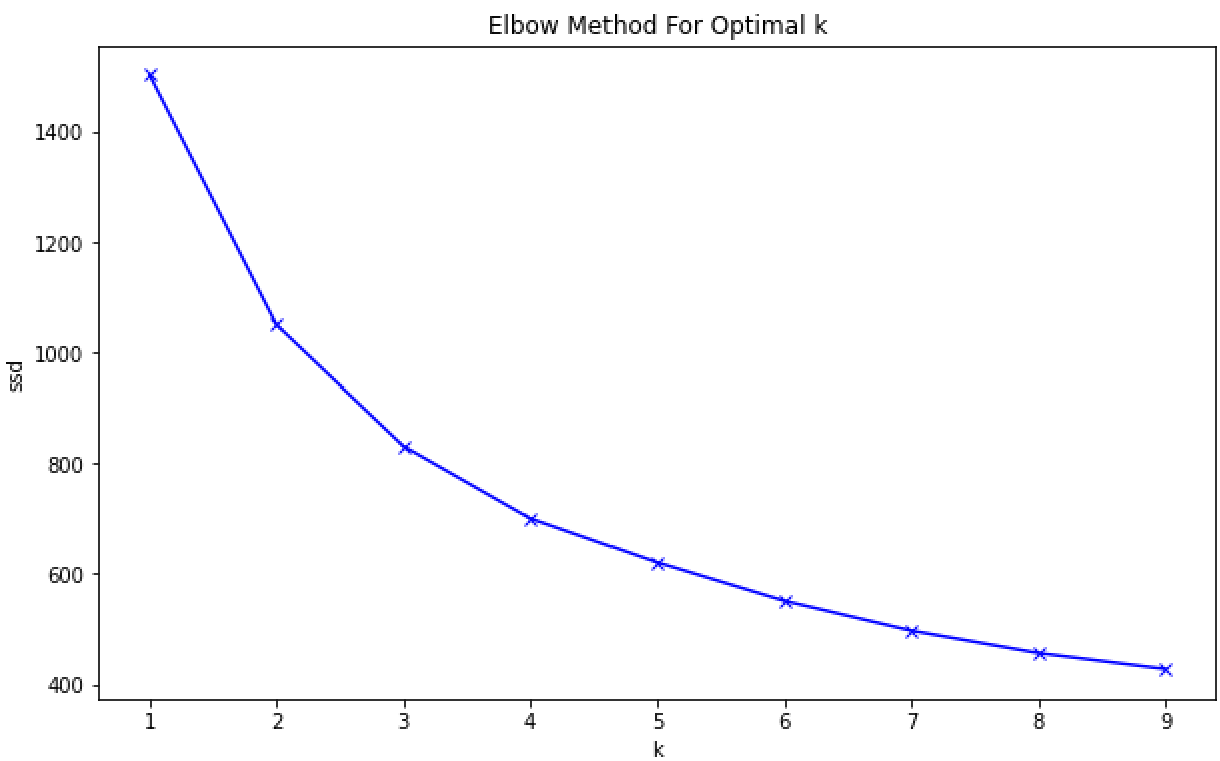
```
#Get visualization for child mortality vs. health data for the various countries
exp.scatter(data_frame=df,x = 'child_mort',y = 'health',color='country')
```

7.0 K-Means Clustering

```
#Elbow plot
from sklearn.cluster import KMeans

ssd = []
K = range(1,10)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(data)
    ssd.append(km.inertia_)
```

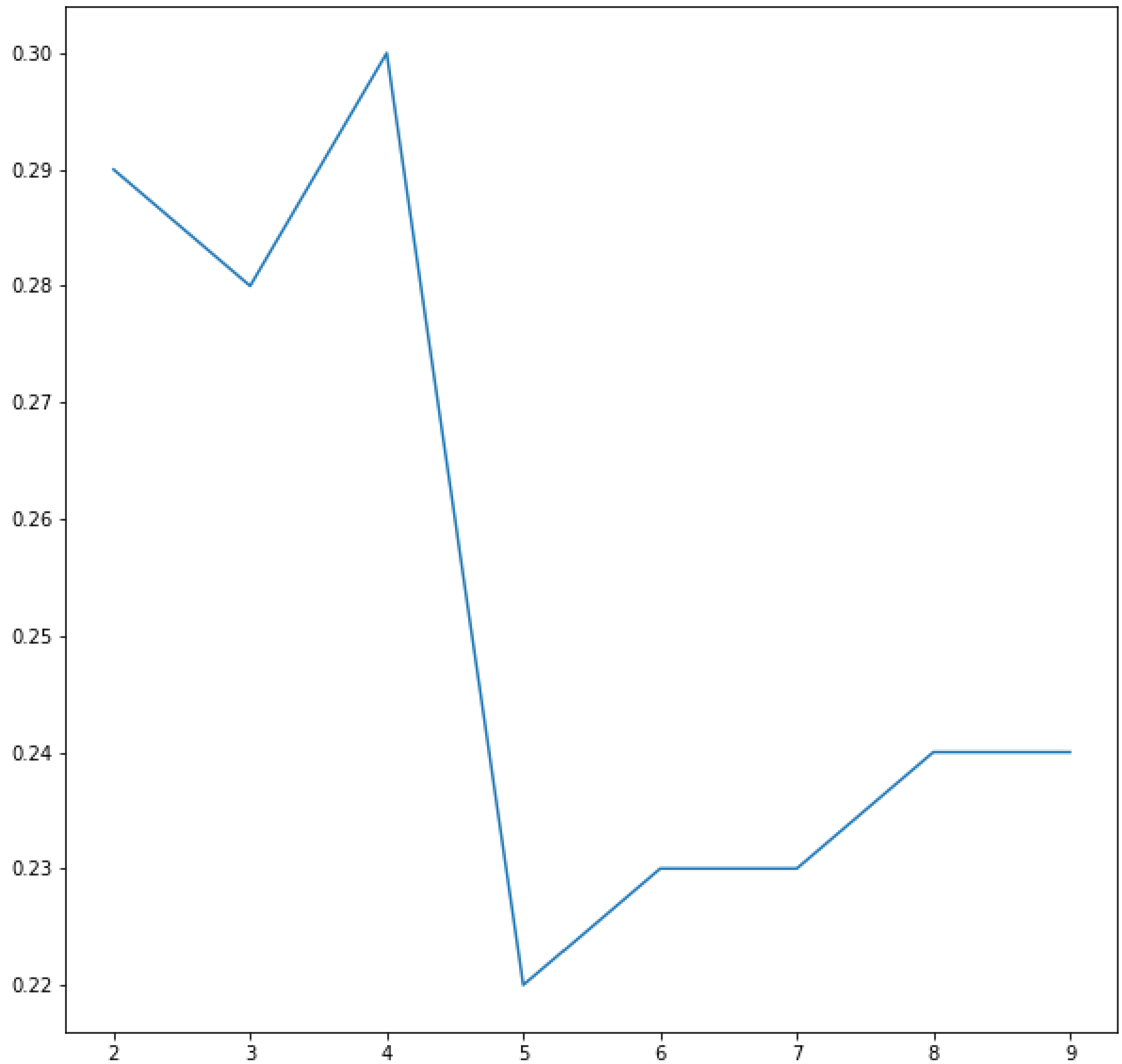
```
plt.figure(figsize=(10,6))
plt.plot(K, ssd, 'bx-')
plt.xlabel('k')
plt.ylabel('ssd')
plt.title('Elbow Method For Optimal k')
plt.show()
```



Because of the substantial difference errors in the elbow plot, k=3,4 is a good choice. To choose between the two, we choose with silhouette score.

```
#Finding the optimum K to cluster the data set using the silhouette score.
score=[]
plt.figure(figsize=(10,10))
for i in range(2,10):
    k=KMeans(i)
    k.fit(data)
    score.append(np.round(silhouette_score(data,k.labels_),2))
plt.plot(range(2,10),score)
```

```
[<matplotlib.lines.Line2D at 0x7fdc10fb7990>]
```



According to the graph above, k=4 has a higher silhouette score than k=3. Although however, k=5 shows a high silhouette score, but the number of clusters should not be too high. Thus we choose k=4.

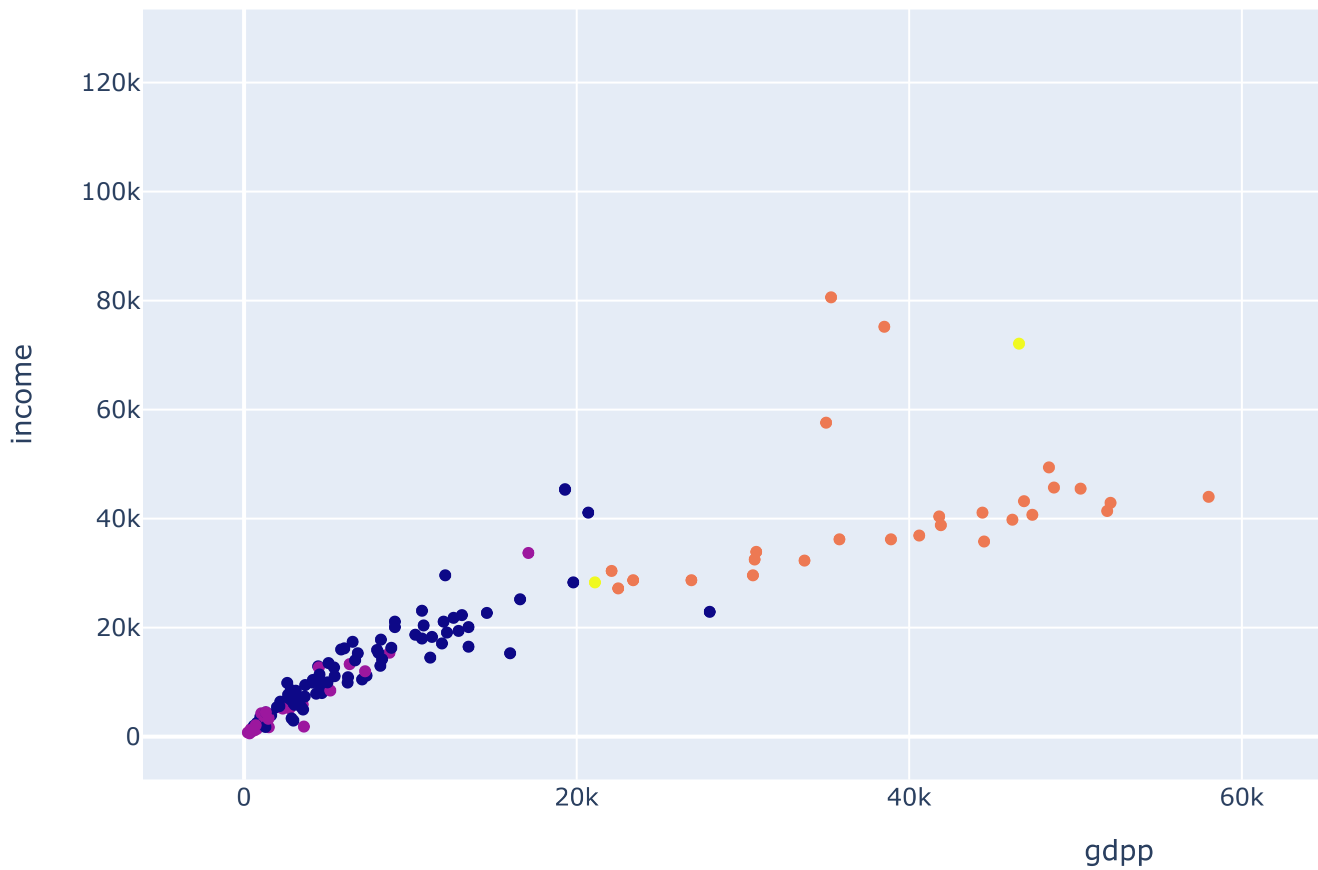
```
k=KMeans(n_clusters=4,random_state=42)
k.fit(data)
```

```
df['k_labels']=k.labels_

pred = k.labels_
print(pred)
```

```
[1 0 0 1 0 0 0 2 2 0 0 0 0 0 0 2 0 1 0 0 0 1 0 2 0 1 1 0 1 2 0 1 1 0 0 0 1
 1 1 0 1 0 2 0 2 0 0 0 0 1 1 0 0 2 2 1 1 0 2 1 2 0 0 1 1 0 1 0 2 0 0 0 1 2
 2 2 0 2 0 0 1 1 2 0 1 0 0 1 1 0 0 3 0 1 1 0 0 1 3 1 0 0 0 0 0 0 1 0 1 0 2
 2 1 1 2 0 1 0 0 0 0 0 2 2 0 0 1 0 0 1 0 0 1 3 0 2 0 1 2 2 0 0 1 0 2 2 0 1
 0 1 1 0 0 0 0 1 0 2 2 2 0 0 0 0 0 1 1]
```

```
#Cluster GDP vs. Income
exp.scatter(data_frame= df,x = 'gdpp',y = 'income',color=k.labels_)
```



We can observe that as GDP increases, income goes up as well. When a country's GDP rises, its people's living standards go up with it.

8.0 Visualize the Clusters with PCA

Principal Component Analysis (PCA) is an unsupervised, non-parametric statistical technique primarily used for dimensionality reduction in machine learning. High dimensionality means that the dataset has a large number of features. PCA can also be used to filter noisy datasets, such as image compression. Source: [Medium.]

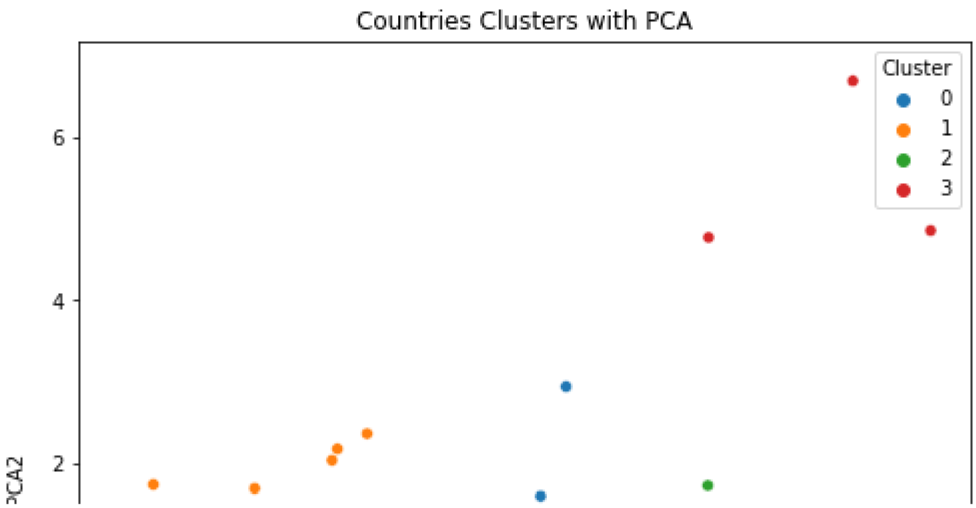
```
#Perform linear dimensionality reduction on data using PCA
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca_model = pca.fit_transform(data)
data_transform = pd.DataFrame(data = pca_model, columns = ['PCA1', 'PCA2'])
data_transform['Cluster'] = pred
```

```
#Get the transformed data
data_transform.head()
```

	PCA1	PCA2	Cluster
0	-2.913025	0.095621	1
1	0.429911	-0.588156	0
2	-0.285225	-0.455174	0
3	-2.932423	1.695555	1
4	1.033576	0.136659	0

```
#Using PCA, visualize country clusters.
plt.figure(figsize=(8,8))
g = sns.scatterplot(data=data_transform, x='PCA1', y='PCA2', palette=sns.color_pa
title = plt.title('Countries Clusters with PCA')
```



We have successfully used KMeans and PCA analysis to decompose our dataset into clusters. However, having grouped the countries into clusters, we still do not know which countries need help the most. Therefore, to correctly classify these countries, we'll apply a different algorithm called hierarchical clustering, which will help us group them according to their level of need.

9.0 Hierarchial Clustering

We'll utilise hierarchical clustering to figure out which countries require assistance.

```
from sklearn.cluster import AgglomerativeClustering
```

```
score=[]
for i in range(2,10):
    a=AgglomerativeClustering(i)
    a.fit(data)
    score.append(np.round(silhouette_score(data,a.labels_),2))
plt.plot(range(2,10),score)
```

```
[<matplotlib.lines.Line2D at 0x7fdc10e93d50>]
```

- The silhouette score of k=2 is good.

```
#Cluster the data into two clusters using Agglomerative clustering.
a=AgglomerativeClustering(2)
a.fit(data)
df['hier_labels']=a.labels_
```

```
#The "hier_labels" is added to dataframe.
df.head()
```

	country	child_mort	exports	health	imports	income	inflation	life_expec	tota
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	
4	Antigua and	10.3	45.5	6.03	58.9	19100	1.44	76.8	

```
df.drop('hier_labels',axis=1).groupby(['k_labels','country']).mean()
```

Replace labels with 'Needs help' or 'Is Self-sufficient'

```
def func(x):
    if x==0:
        return 'Needs help priority-1'
    elif x==1:
        return 'Needs help priority-2'
    elif x==2:
        return 'Needs help priority-3'
    else:
        return 'Is Self-sufficient'
df['k_labels']=df['k_labels'].map(lambda x: func(x))
```

```
df.drop('k_labels',axis=1).groupby(['hier_labels','country']).mean()
```

		child_mort	exports	health	imports	income	inflati
hier_labels	**country**						
0	**Afghanistan**	90.2	10.0	7.58	44.9	1610.0	9.4
	Albania	16.6	28.0	6.55	48.6	9930.0	4.4
	Algeria	27.3	38.4	4.17	31.4	12900.0	16.1
	Angola	119.0	62.3	2.85	42.9	5900.0	22.4
	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100.0	1.4
...	**...**	...	...	...	...	...	
1	**Sweden**	3.0	46.2	9.63	40.7	42900.0	0.9
	Switzerland	4.5	64.0	11.50	53.3	55500.0	0.3
	United Arab Emirates	8.6	77.7	3.66	63.6	57600.0	12.5
	United Kingdom	5.2	28.2	9.64	30.8	36200.0	1.5
	United States	7.3	12.4	17.90	15.8	49400.0	1.2

167 rows × 9 columns

```
def func(x):
    if x==0:
        return 'Needs help'
    else:
        return 'Is Self-sufficient'
df['hier_labels']=df['hier_labels'].map(lambda x: func(x))
```

```
df.head(20)
```

	country	child_mort	exports	health	imports	income	inflation	life_expec	tot
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.440	56.2	
1	Albania	16.6	28.0	6.55	48.6	9930	4.490	76.3	
2	Algeria	27.3	38.4	4.17	31.4	12900	16.100	76.5	
3	Angola	119.0	62.3	2.85	42.9	5900	22.400	60.1	
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.440	76.8	
5	Argentina	14.5	18.9	8.10	16.0	18700	20.900	75.8	
6	Armenia	18.1	20.8	4.40	45.3	6700	7.770	73.3	
7	Australia	4.8	19.8	8.73	20.9	41400	1.160	82.0	
8	Austria	4.3	51.3	11.00	47.8	43200	0.873	80.5	
9	Azerbaijan	39.2	54.3	5.88	20.7	16000	13.800	69.1	

10.0 Result

```
print('Based on K-Means clustering, the countries that require the most assistance
df.loc[df['k_labels']=='Needs help priority-1']['country'].to_list()
```

```
'Fiji',
'Georgia',
'Grenada',
'Guatemala',
'Guyana',
```

```
'Hungary',
'India',
'Indonesia',
'Iran',
'Jamaica',
'Jordan',
'Kazakhstan',
'Kyrgyz Republic',
'Latvia',
'Lebanon',
'Libya',
'Lithuania',
'Macedonia, FYR',
'Malaysia',
'Maldives',
'Mauritius',
'Micronesia, Fed. Sts.',
'Moldova',
'Mongolia',
'Montenegro',
'Morocco',
'Myanmar',
'Nepal',
'Oman',
'Panama',
'Paraguay',
'Peru',
'Philippines',
'Poland',
'Romania',
'Russia',
'Samoa',
'Saudi Arabia',
'Serbia',
'Seychelles',
'Slovak Republic',
'Solomon Islands',
'Sri Lanka',
'St. Vincent and the Grenadines',
'Suriname',
'Tajikistan',
'Thailand',
'Tonga',
'Tunisia',
'Turkey',
'Turkmenistan',
'Ukraine',
'Uruguay',
'Uzbekistan',
'Vanuatu',
'Venezuela',
'Vietnam']
```

```
print('Based on Hierarchial clustering, the countries that require the most assis
```

```
df.loc[df['hier_labels']=='Needs help']['country'].to_list()
```

```
 'Malawi',
 'Malaysia',
 'Maldives',
 'Mali',
 'Mauritania',
 'Mauritius',
 'Micronesia, Fed. Sts.',
 'Moldova',
 'Mongolia',
 'Montenegro',
 'Morocco',
 'Mozambique',
 'Myanmar',
 'Namibia',
 'Nepal',
 'Niger',
 'Nigeria',
 'Pakistan',
 'Panama',
 'Paraguay',
 'Peru',
 'Philippines',
 'Poland',
 'Romania',
 'Russia',
 'Rwanda',
 'Samoa',
 'Senegal',
 'Serbia',
 'Seychelles',
 'Sierra Leone',
 'Slovak Republic',
 'Slovenia',
 'Solomon Islands',
 'South Africa',
 'South Korea',
 'Sri Lanka',
 'St. Vincent and the Grenadines',
 'Sudan',
 'Suriname',
 'Tajikistan',
 'Tanzania',
 'Thailand',
 'Timor-Leste',
 'Togo',
 'Tonga',
 'Tunisia',
 'Turkey',
 'Turkmenistan',
 'Uganda',
 'Ukraine',
 'Uruguay',
 'Uzbekistan',
 'Vanuatu',
 'Venezuela',
```

```
 'Venezuela',
 'Vietnam',
 'Yemen',
 'Zambia']
```

10.1 Summary

- From the preceding, we are recommending this machine learning model to "HELP International" to assist with their humanitarian work because it has correctly clustered the countries according to the level of their needs. Our client now knows which countries need the most help, and they now know how to apportion the 10 million dollar largesse.
- To further strengthen the model, we recommend that "Help International" consults with a subject matter expert who may have additional need-based suggestions for the various countries. It would help with modeling the data to increase model performance and ensure that the algorithms employed and the results obtained are equitable for all nations represented. It is so that the model developed isn't biased in favor of any country or group.

References

Help International --- https://help-international.org/

PCA: Application in Machine Learning --- https://bit.ly/3skwHv9

Exploratory Data Analysis --- https://bit.ly/35NMyuB