Clustering Assignment

Submitted by – Sourabh Shrivas

Objective: HELP International humanitarian NGO, committed to fight poverty and provide the people of backward countries with basic amenities and relief during the time of disasters and natural calamities. We run a lot of operational projects from time to time, along with advocacy, drives to raise awareness as well as for funding purposes.

Problem statement: During the recent funding programmed, we have been able to raise around $ 10 million. As an analyst, we have to come up with the countries list that are in the direst need of aid.

Approach used

We used both techniques :

1 – K means clustering

2 – Hierarchical clustering

```
# Read the given CSV file, and view some sample records
df_country = pd.read_csv("Country-data.csv")
print(df_country.shape)
df_country.head()
```

```
(167, 10)
```

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200

Data Inspections and Quality checks:

```
df_country.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 10 columns):
country       167 non-null object
child_mort    167 non-null float64
exports       167 non-null float64
health        167 non-null float64
imports       167 non-null float64
income        167 non-null int64
inflation     167 non-null float64
life_expec    167 non-null float64
total_fer     167 non-null float64
gdpp          167 non-null int64
dtypes: float64(7), int64(2), object(1)
memory usage: 13.1+ KB
```

Observations:

- Index range is from 0 to 167 in a sequence integer number
- Data types looks correct.
- We do not have any null values in dataframe.

```
missing_values_table(df_country)
```

```
Given dataframe has 10 columns and 167 Rows.
There are 0 columns that have missing values.
```

	null value %	Missing Values

```
#checking for duplicates
df_country.duplicated(subset = ['country'], keep = False).sum()
```

```
0
```

Converting exports, imports and health into there actual values.

```
#Converting exports, imports and health into there actual values.
df_country['exports'] = df_country['exports']*df_country['gdpp']/100
df_country['health'] = df_country['health']*df_country['gdpp']/100
df_country['imports'] = df_country['imports']*df_country['gdpp']/100

df_country.head()
```

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	55.30	41.9174	248.297	1610	9.44	56.2	5.82	553
1	Albania	16.6	1145.20	267.8950	1987.740	9930	4.49	76.3	1.65	4090
2	Algeria	27.3	1712.64	185.9820	1400.440	12900	16.10	76.5	2.89	4460
3	Angola	119.0	2199.19	100.6050	1514.370	5900	22.40	60.1	6.16	3530
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100	1.44	76.8	2.13	12200

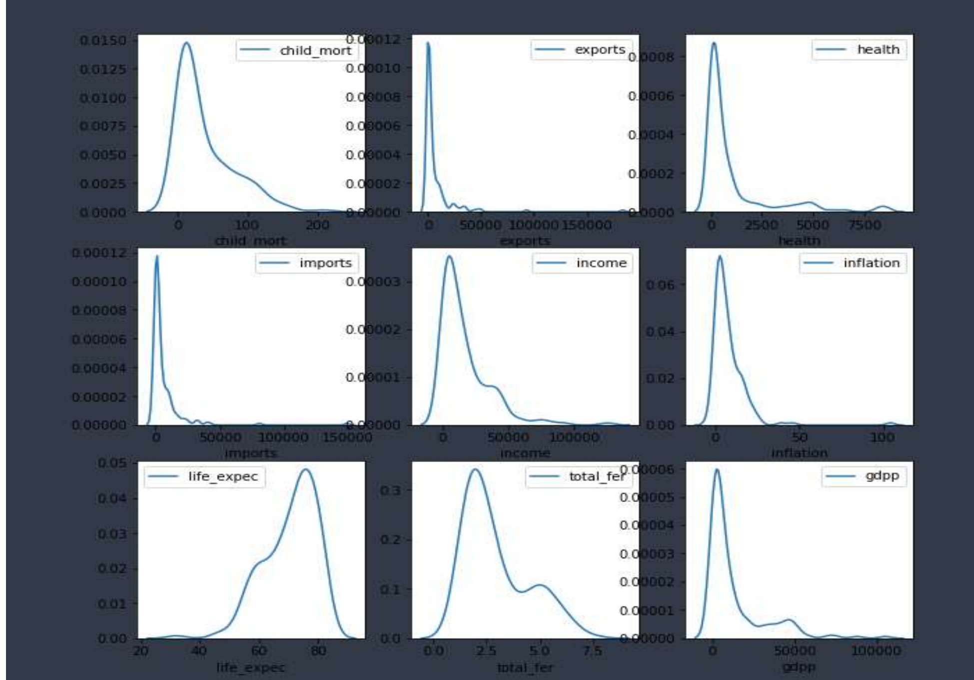
EDA:

- We will be using all the columns for clustering and only 3 columns namely GDPP, CHILD_MORT and INCOME for profiling.
- In first observation, Seems like none of the numeric variables are helping for creating the profile for cluster.

Univariate Analysis:

Continuous Variables –

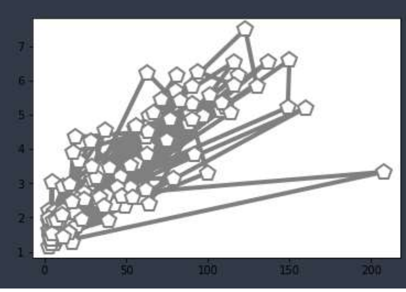
```
plt.figure(figsize = (10,10))

for i in enumerate(list_of_columns[1:]):
    plt.subplot(3,3,i[0]+1)
    sns.distplot(df_country[i[1]], hist = False,label=i[1])
```



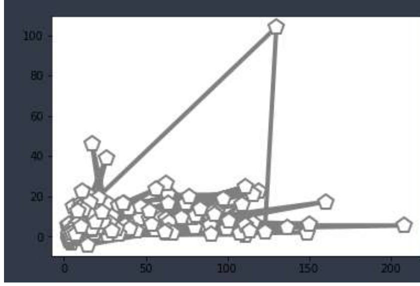
Bivariate Analysis:

Analysis for Bivariate - Continuous - Continuous

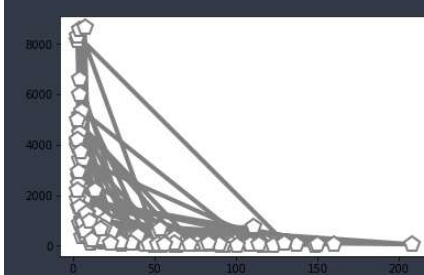
```
plt.plot(df_country.child_mort, df_country.total_fer, '-p', color='gray',
         markersize=15, linewidth=4,
         markerfacecolor='white',
         markeredgecolor='gray',
         markeredgewidth=2)
plt.show()
```




```
plt.plot(df_country.child_mort, df_country.inflation, '-p', color='gray',
         markersize=15, linewidth=4,
         markerfacecolor='white',
         markeredgecolor='gray',
         markeredgewidth=2)
plt.show()
```

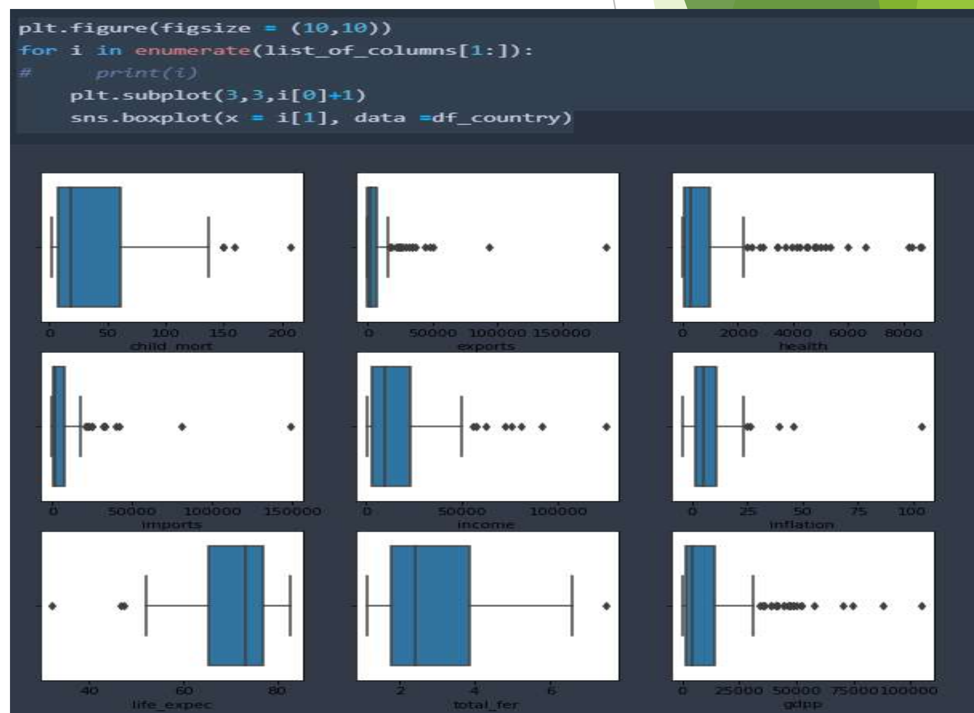



```
plt.plot(df_country.child_mort, df_country.health, '-p', color='gray',
         markersize=15, linewidth=4,
         markerfacecolor='white',
         markeredgecolor='gray',
         markeredgewidth=2)
plt.show()
```



Outliers Observations and Treatment:

- gdpp, income and inflation columns are having high outliers.
- let's not remove outlier from inflation as this might lead to loss in country details which are not doing well- socio-economically(countries with direst need of aid).
- Outliers in lower range represents those countries having low income, so we may loose those countries by dropping them
- child mortality has higher range that means low income for the country.



Capping the data to take care outliers:

```python
# winsorizing outliers/capping outliers
def percentile_capping(df, cols,from_low_end,from_high_end):
    for col in cols:
        lower_bound = df[col].quantile(from_low_end)
        upper_bound = df[col].quantile(1-from_high_end)

        df[col] = np.where(df[col]>upper_bound,  upper_bound,
                           np.where(df[col]<lower_bound, lower_bound,df[col]))
```

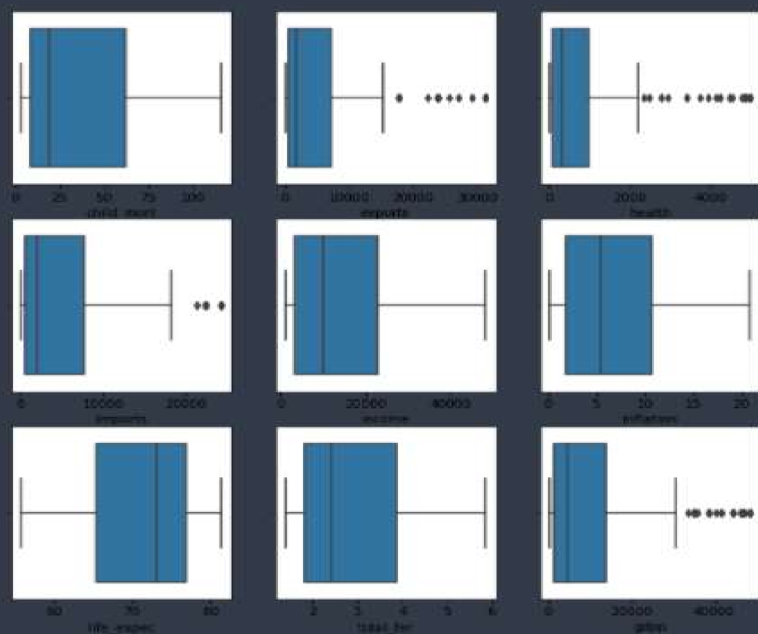
```python
percentile_capping(df_country_cap,df_country_cap.columns[1:],0.05,0.05)
```

```python
df_country.describe()
```

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
count	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000
mean	38.270060	7420.618847	1056.733204	6588.352108	17144.688623	7.781832	70.555689	2.947964	12964.155689
std	40.328931	17973.885795	1801.408906	14710.810418	19278.067698	10.570704	8.893172	1.513848	18328.704809
min	2.600000	1.076920	12.821200	0.651092	609.000000	-4.210000	32.100000	1.150000	231.000000
25%	8.250000	447.140000	78.535500	640.215000	3355.000000	1.810000	65.300000	1.795000	1330.000000
50%	19.300000	1777.440000	321.886000	2045.580000	9960.000000	5.390000	73.100000	2.410000	4660.000000
75%	62.100000	7278.000000	976.940000	7719.600000	22800.000000	10.750000	76.800000	3.880000	14050.000000
max	208.000000	183750.000000	8663.600000	149100.000000	125000.000000	104.000000	82.800000	7.490000	105000.000000

Outliers after capping on the data:

```
plt.figure(figsize = (10,10))
for i in enumerate(list_of_columns[1:]):
#     print(i)
    plt.subplot(3,3,i[0]+1)
    sns.boxplot(x = i[1], data =df_country_cap)
```



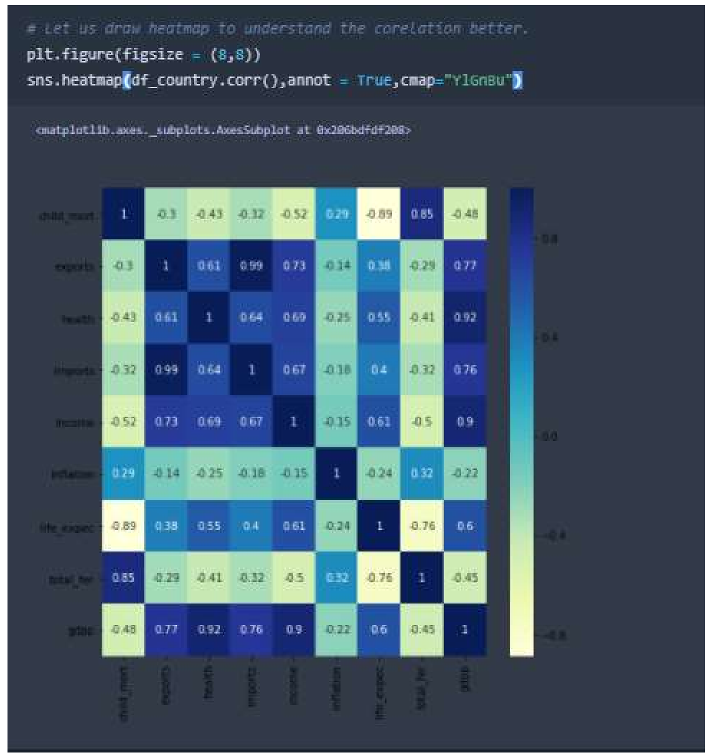
- Still gdpp and health columns looks skewed, we can increared the upper bound limit for there columns to take care of these columns. for now moving ahead with our analysis.

Data Visualization:

Data Correlation:

High correlation :

- between total_fer and child_mort
- between gdpp and income, and
- between imports and exports



Scaling:

5: Scaling

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
df_country_scale = ss.fit_transform(df_country_cap[df_country_cap.columns[1:]])
df_country_scale
```

```
array([[ 1.47958789, -0.66803864, -0.62977844, ..., -1.82530988,
         2.02071786, -0.7578739 ],
       [-0.56002364, -0.54238923, -0.47380714, ...,  0.68245351,
        -0.88733091, -0.52377507],
       [-0.26350403, -0.47604845, -0.53034406, ...,  0.70740638,
        -0.02258739, -0.49928636],
       ...,
       [-0.37435248, -0.56600553, -0.59686482, ...,  0.2832076 ,
        -0.67811877, -0.70777132],
       [ 0.54014725, -0.63033074, -0.61187406, ..., -0.41547275,
         1.21873798, -0.70777132],
       [ 1.28283189, -0.61312123, -0.59935647, ..., -1.8777109 ,
         1.72782086, -0.69784347]])
```

```
df_country_scale = pd.DataFrame(df_country_scale,columns = df_country_cap.columns[1:])
# df_country_scale.columns = df_country.columns[1:]
df_country_scale.head()
```

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	1.479588	-0.668039	-0.629778	-0.733291	-0.960575	0.387667	-1.825310	2.020718	-0.757874
1	-0.560024	-0.542389	-0.473807	-0.472674	-0.395590	-0.404004	0.682454	-0.887331	-0.523775
2	-0.263504	-0.476048	-0.530344	-0.560668	-0.193907	1.452825	0.707406	-0.022587	-0.499286
3	2.194560	-0.419165	-0.589272	-0.543598	-0.669255	2.215708	-1.338729	2.049310	-0.560839
4	-0.734610	-0.027297	-0.150953	0.306143	0.227115	-0.891802	0.744836	-0.552591	0.012991

Hopkin Tendency:

4: HOPKINS: Cluster Tendency before scaling (Hopkins Statistics)

```python
#Calculating the Hopkins statistic
from sklearn.neighbors import NearestNeighbors
from random import sample
from numpy.random import uniform
import numpy as np
from math import isnan

def hopkins(X):
    d = X.shape[1]
    #d = len(vars) # columns
    n = len(X) # rows
    m = int(0.1 * n)
    nbrs = NearestNeighbors(n_neighbors=1).fit(X.values)

    rand_X = sample(range(0, n, 1), m)

    ujd = []
    wjd = []
    for j in range(0, m):
        u_dist, _ = nbrs.kneighbors(uniform(np.amin(X,axis=0),np.amax(X,axis=0),d).reshape(1, -1), 2, r
        ujd.append(u_dist[0][1])
        w_dist, _ = nbrs.kneighbors(X.iloc[rand_X[j]].values.reshape(1, -1), 2, return_distance=True)
        wjd.append(w_dist[0][1])

    H = sum(ujd) / (sum(ujd) + sum(wjd))
    if isnan(H):
        print(ujd, wjd)
        H = 0

    return H
```

```python
hopkins(df_country[df_country_cap.columns[1:]])
```

```
0.9511513752482844
```

Find the best value of K = 3:

silhouette score:

7.1 - silhouette_score

```python
from sklearn.metrics import silhouette_score
# looks at the maximum value
ss = []
for k in range(2,11):
    kmeans = KMeans(n_clusters = k).fit(df_country_scale)
    ss.append([k, silhouette_score(df_country_scale, kmeans.labels_)])

sil = pd.DataFrame(ss)
plt.plot(sil[0],sil[1])
```

```
[<matplotlib.lines.Line2D at 0x206c170bef0>]
```

elbow curve:

7.2 - elbow curve

```python
# ssd elbow curve
ssd = []
for k in range(2,11):
    kmean = KMeans(n_clusters = k).fit(df_country_scale)
    ssd.append([k, kmean.inertia_])

ssd = pd.DataFrame(ssd)
plt.plot(ssd[0], ssd[1])
```

```
[<matplotlib.lines.Line2D at 0x206c15a99b0>]
```

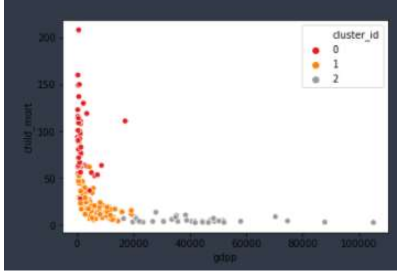
Final KMeans with K==3

Plotting the clusters through Scatter plot:

GDPP, Child Mort and and income

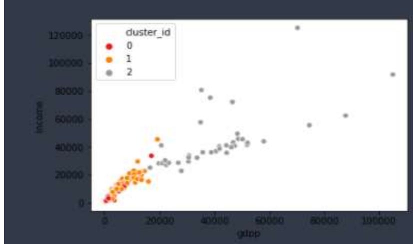
```
sns.scatterplot(x = 'gdpp',
                y ='child_mort',
                hue = 'cluster_id',
                data =df_country_kmeans, palette = 'Set1')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x206c179d6d8>
```



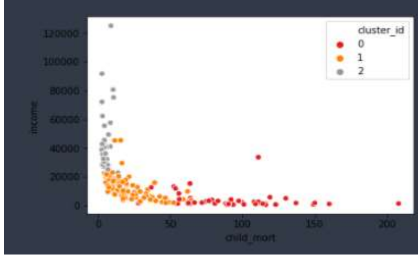

```
sns.scatterplot(x = 'gdpp',
                y ='income',
                hue = 'cluster_id',
                data =df_country_kmeans,
                palette = 'Set1')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x206c1979128>
```



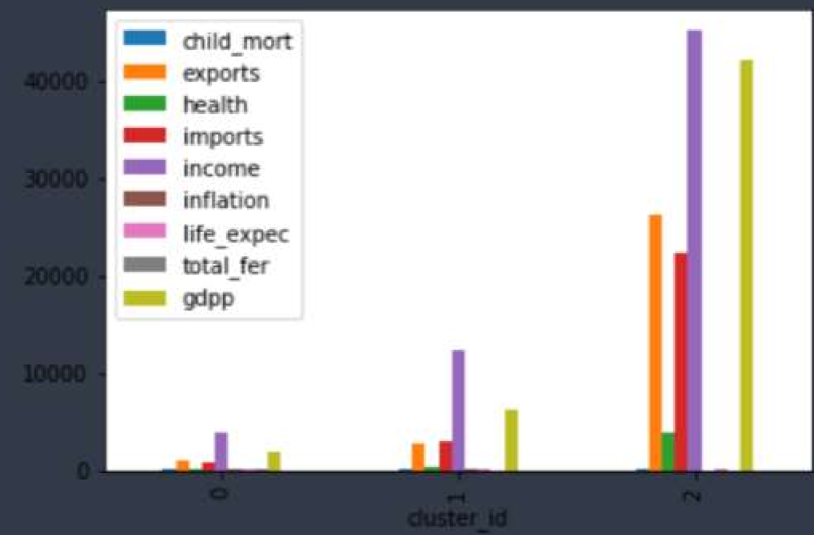

```
sns.scatterplot(x = 'child_mort',
                y ='income',
                hue = 'cluster_id',
                data =df_country_kmeans,
                palette = 'Set1')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x206c15ece10>
```

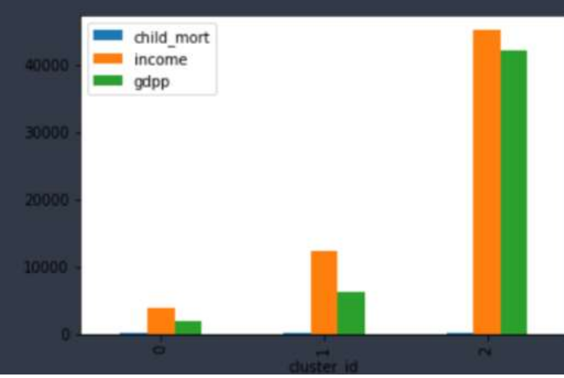


Making sense out of Cluster:

```
# Making sense out of the clsuters
df_country_kmeans.drop('country', axis = 1).groupby('cluster_id').mean().plot(kind = 'bar')
plt.show()
```




```
# GDP, INCOME AND CHID_MORT
df_country_kmeans.drop(['exports',
                 'health',
                 'imports',
                 'inflation',
                 'life_expec',
                 'total_fer'], axis = 1).groupby('cluster_id').mean().plot(kind = 'bar')

plt.show()
```



Countries, which are direst need of aid : (K-Means)

As per K- means clustering, and based on gdpp, child_mort and income cluster, following are the countries, which are direst need of aid:=

1 - Haiti

2 - Sierra Leone

3 - Chad

4 - Central African Republic

5 - Mali

6 - Nigeria

7 - Niger

8 - AngolaCongo, Dem. Rep.

9 - Congo, Dem. Rep.

10 - Burkina Faso

```
df_country_kmeans[df_country_kmeans['cluster_id'] == 0].sort_values(by = ['child_mort',
                                                                          'income',
                                                                          'gdpp'],
                                                                ascending = [False, True,True]).head(10)
```

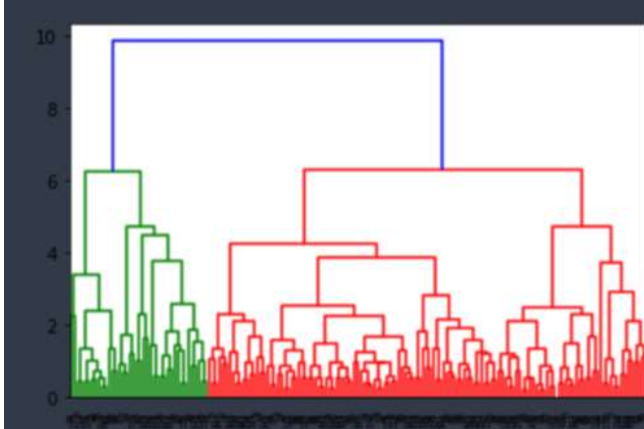
	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster_id
66	Haiti	208.0	101.286	45.7442	428.314	1500	5.45	32.1	3.33	662	0
132	Sierra Leone	160.0	67.032	52.2690	137.655	1220	17.20	55.0	5.20	399	0
32	Chad	150.0	330.096	40.6341	390.195	1930	6.39	56.5	6.59	897	0
31	Central African Republic	149.0	52.628	17.7508	118.190	888	2.01	47.5	5.21	446	0
97	Mali	137.0	161.424	35.2584	248.508	1870	4.37	59.5	6.55	708	0
113	Nigeria	130.0	589.490	118.1310	405.420	5150	104.00	60.5	5.84	2330	0
112	Niger	123.0	77.256	17.9568	170.868	814	2.55	58.8	7.49	348	0
3	Angola	119.0	2199.190	100.6050	1514.370	5900	22.40	60.1	6.16	3530	0
37	Congo, Dem. Rep.	116.0	137.274	26.4194	165.664	609	20.80	57.5	6.54	334	0
25	Burkina Faso	116.0	110.400	38.7550	170.200	1430	6.81	57.9	5.87	575	0

Hierarchical Clustering:

Also as per Hierarchical clustering , and based on gdpp, child_mort and income cluster, following are the countries, which are direst need of aid:

1 - Haiti
2 - Sierra Leone
3 - Chad
4 - Central African Republic
5 - Mali
6 - Nigeria
7 - Niger
8 – Angola Congo, Dem. Rep.
9 - Congo, Dem. Rep.
10 - Burkina Faso

```
#single linkage
mergings = linkage(df_country_scale,
                   method = 'complete',
                   metric = 'euclidean')

dendrogram(mergings)
plt.show()
```



Countries, which are direst need of aid : (Hierarchical Clustering)

```
df_country_kmeans_2[df_country_kmeans_2['cluster_id'] == 0].sort_values(by = ['child_mort',
                                                                  'income',
                                                                  'gdpp'],
                                                              ascending = [False, True,True]).head(10)
```

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster_id
66	Haiti	208.0	101.286	45.7442	428.314	1500	5.45	32.1	3.33	662	0
132	Sierra Leone	160.0	67.032	52.2690	137.655	1220	17.20	55.0	5.20	399	0
32	Chad	150.0	330.096	40.6341	390.195	1930	6.39	56.5	6.59	897	0
31	Central African Republic	149.0	52.628	17.7508	118.190	888	2.01	47.5	5.21	446	0
97	Mali	137.0	161.424	35.2584	248.508	1870	4.37	59.5	6.55	708	0
113	Nigeria	130.0	589.490	118.1310	405.420	5150	104.00	60.5	5.84	2330	0
112	Niger	123.0	77.256	17.9568	170.868	814	2.55	58.8	7.49	348	0
3	Angola	119.0	2199.190	100.6050	1514.370	5900	22.40	60.1	6.16	3530	0
37	Congo, Dem. Rep.	116.0	137.274	26.4194	165.664	609	20.80	57.5	6.54	334	0
25	Burkina Faso	116.0	110.400	38.7550	170.200	1430	6.81	57.9	5.87	575	0