Assignment I

Clustering and Principal Component Analysis

Objective

HELP International is an international humanitarian NGO that is committed to fighting poverty and providing the people of backward countries with basic amenities and relief during the time of disasters and natural calamities. It runs a lot of operational projects from time to time along with advocacy drives to raise awareness as well as for funding purposes.

After the recent funding programmes, they have been able to raise around $ 10 million. Now the CEO of the NGO needs to decide how to use this money strategically and effectively. The significant issues that come while making this decision are mostly related to choosing the countries that are in the direst need of aid.

And this is where you come in as a data analyst. Your job is **to categorize the countries using some socio-economic and health factors that determine the overall development of the country.** Then you need **to suggest the countries which the CEO needs to focus on the most.**

Steps involved are:

1. Read and visualize the data
2. Clean the data
3. Preparing the data
4. Principal Component Analysis
5. Hopkins Statistics
6. K-Means clustering
7. Analyzing the k-means cluster
8. Hierarchical Clustering
9. Analyzing the hierarchical clusters

Step 1: Read and visualize the data

In [1]:
```
# importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# importing dataset
df = pd.read_csv('C:/Users/User/Desktop/practice set/credited assignment/Clustering/Country-data.csv')
df.head()
```

Out[2]:

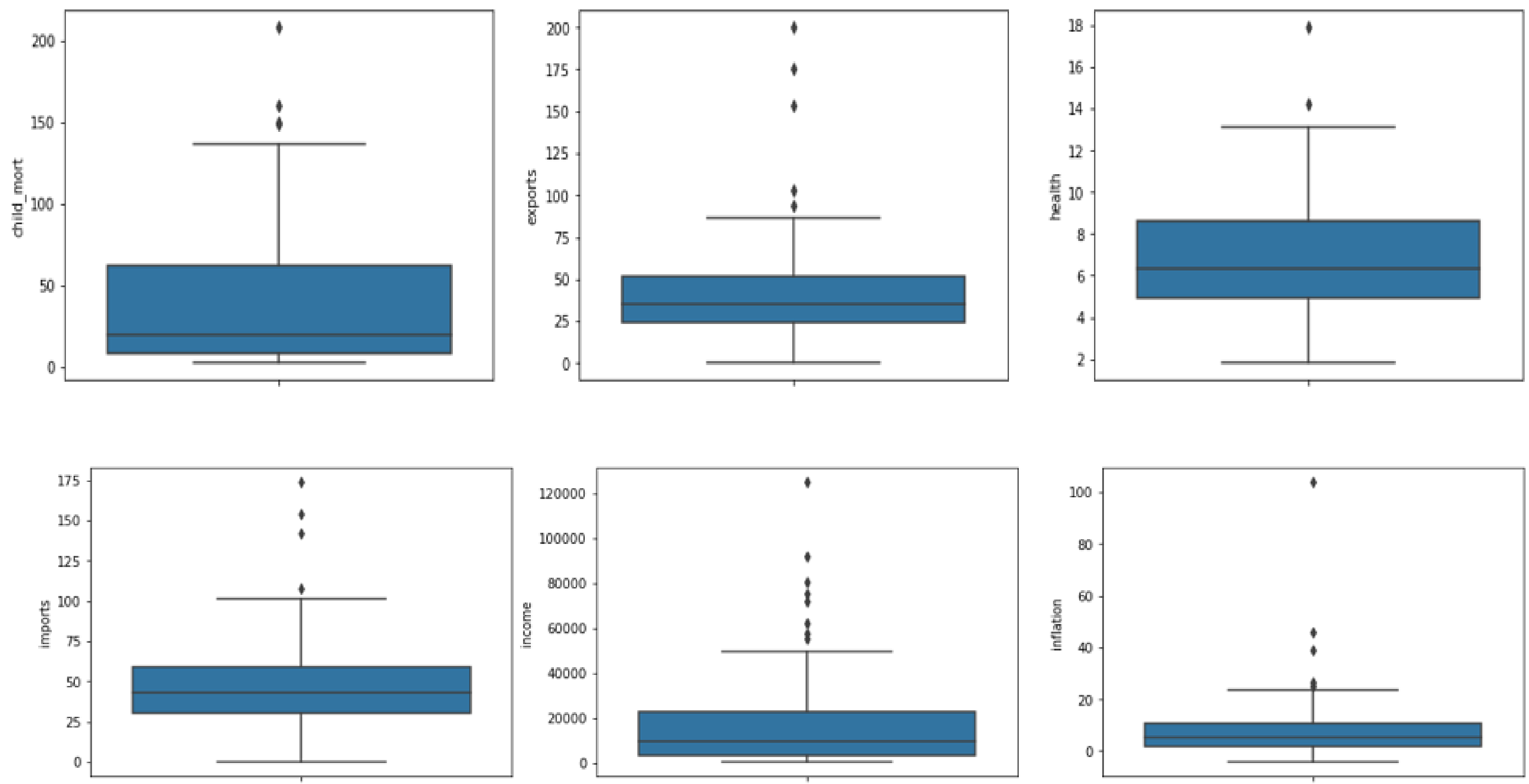
	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1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

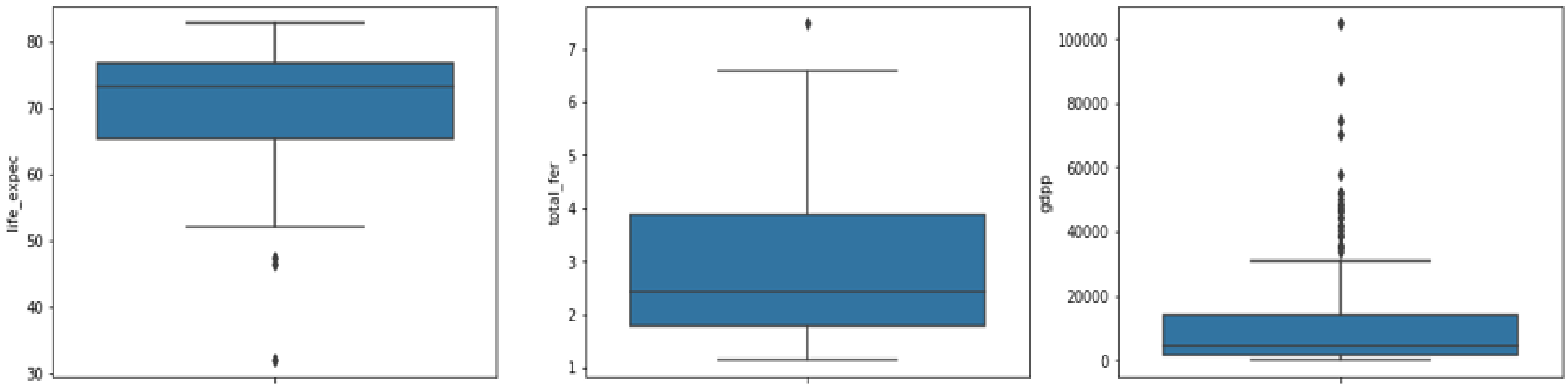
In [3]:
```
# shape of the dataset
df.shape
```

Out[3]: (167, 10)

Step 2: Clean the data

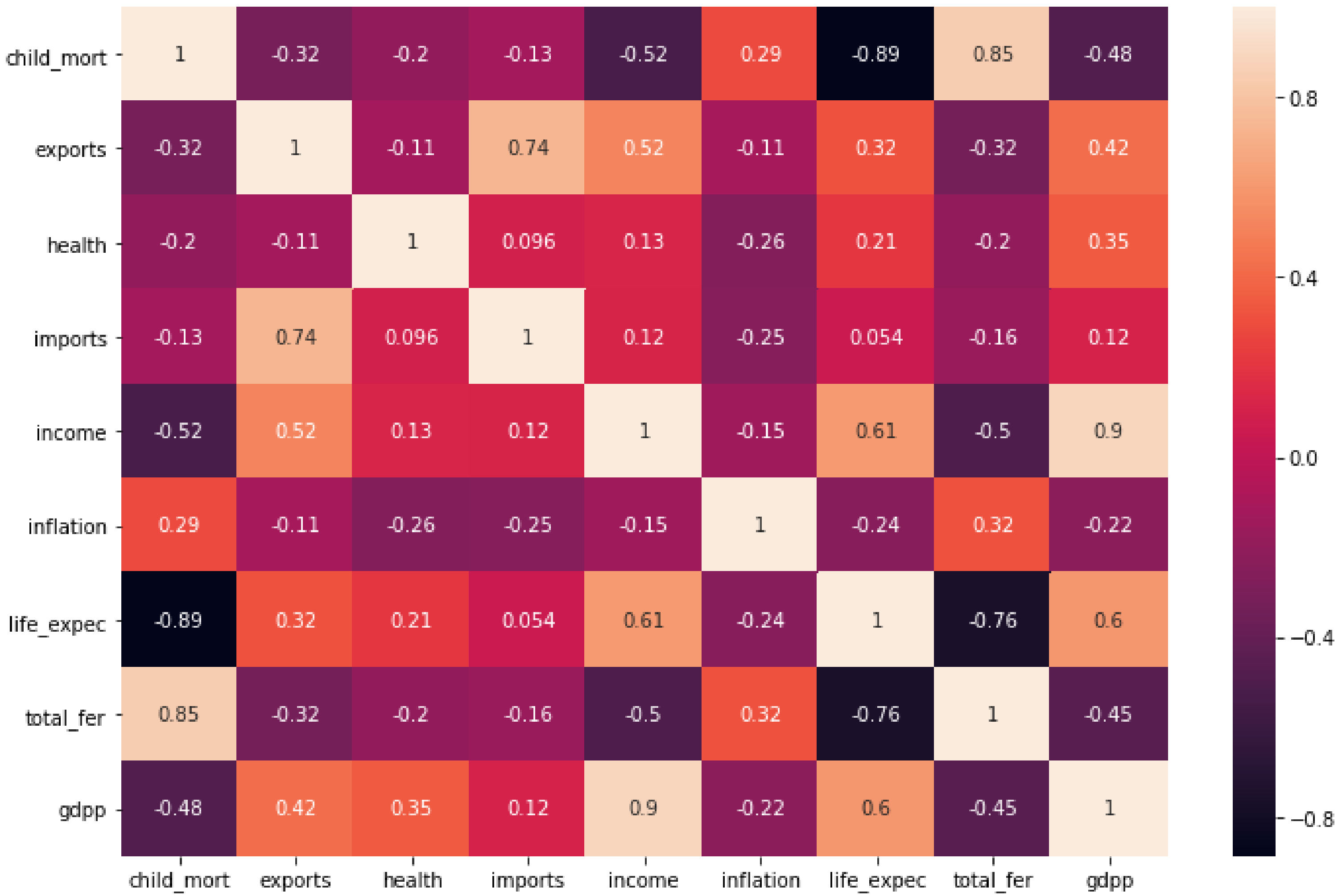
Checking if any **outlier** is present in the dataset





the dataset contain few outliers which can be analysed and treated after Principal Component Analysis

Checking **correlation** among the variables



- child mortality is highly negatively correlated to life_expec
- child mortality is positively correlated to total fertility
- income is negatively correlated to child motality and is highly positively correlated to gdpp

Checking if any **null value** is present

In [9]:
```
# checking if any null value is present
df.isnull().sum()
```

Out[9]:
```
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
```

Formatting variables for better understanding

In [11]:
```
# changing the format of the variables
df['exports'] = df['exports']*df['gdpp']/100
df['health'] = df['health']*df['gdpp']/100
df['imports'] = df['imports']*df['gdpp']/100
```

Step 3: Preparing the data

Dropping country column

In [13]:
```
# dropping the country column
df = df[['child_mort', 'exports', 'health', 'imports', 'income', 'inflation', 'life_expec', 'total_fer', 'gdpp']]
df.head()
```

Out[13]:

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

Scaling the numerical columns

In [14]:
```
# standardization
df=(df-df.mean())/df.std()
df.head()
```

Out[14]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	1.287660	-0.409779	-0.563346	-0.430979	-0.805822	0.156864	-1.614237	1.897176	-0.677143
1	-0.537333	-0.349141	-0.437901	-0.312737	-0.374243	-0.311411	0.645924	-0.857394	-0.484167
2	-0.272015	-0.317571	-0.483372	-0.352660	-0.220182	0.786908	0.668413	-0.038289	-0.463980
3	2.001787	-0.290501	-0.530767	-0.344915	-0.583289	1.382894	-1.175698	2.121770	-0.514720
4	-0.693548	-0.104019	-0.178234	0.040613	0.101427	-0.599944	0.702147	-0.540321	-0.041692

Step 4: Principal Component Analysis

Importing libraries and **doing PCA**

In [15]:
```
# importing the PCA module
from sklearn.decomposition import PCA
pca = PCA(svd_solver='randomized', random_state=42)
```

In [16]:
```
#Doing the PCA on the data
pca.fit(df)
```

Out[16]:
```
PCA(copy=True, iterated_power='auto', n_components=None, random_state=42,
  svd_solver='randomized', tol=0.0, whiten=False)
```

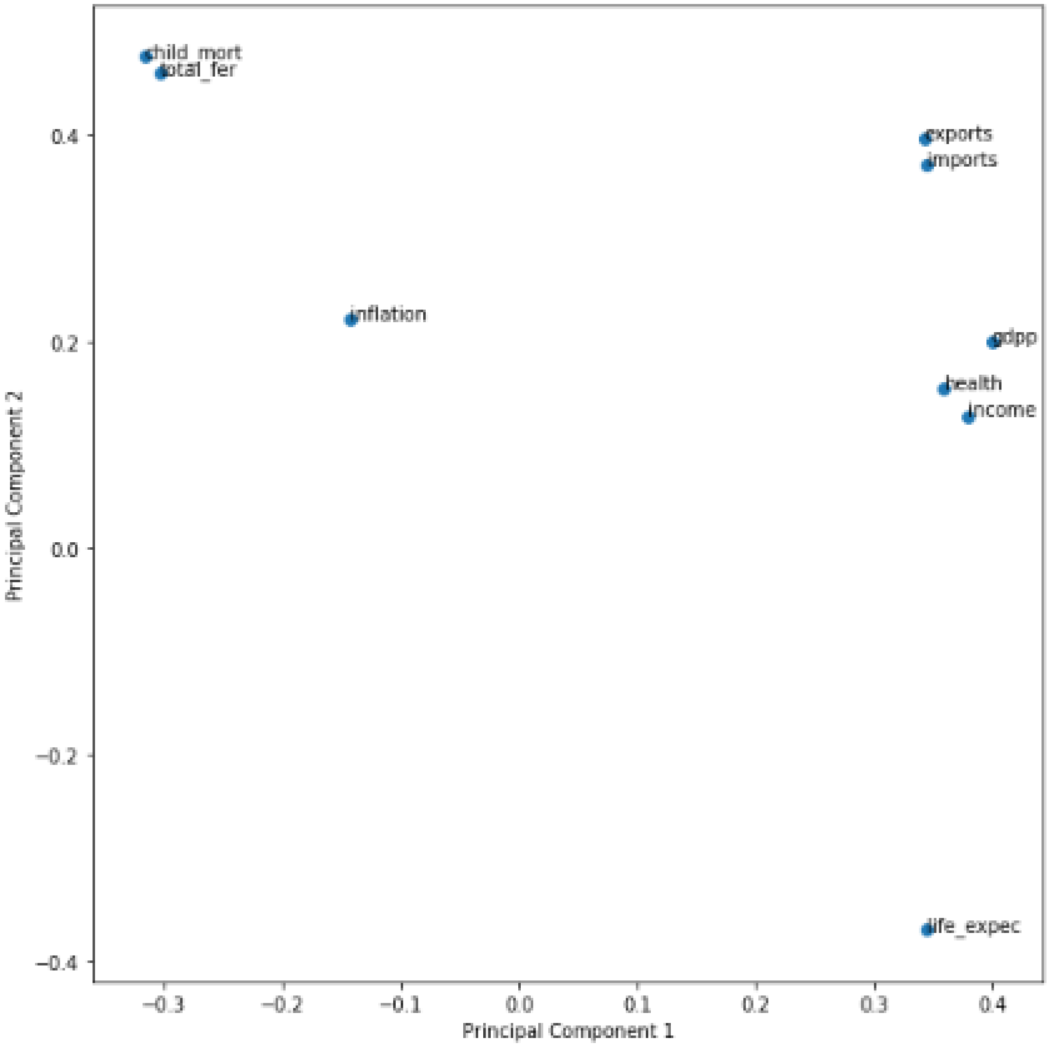
List of principal components

In [18]:
```
colnames = list(df.columns)
pcs_df = pd.DataFrame({'PC1':pca.components_[0],'PC2':pca.components_[1], 'Feature':colnames})
pcs_df.head()
```

Out[18]:

	PC1	PC2	Feature
0	-0.316392	0.476267	child_mort
1	0.342887	0.397311	exports
2	0.358535	0.155053	health
3	0.344865	0.370781	imports
4	0.380041	0.128384	income

Plotting principal components



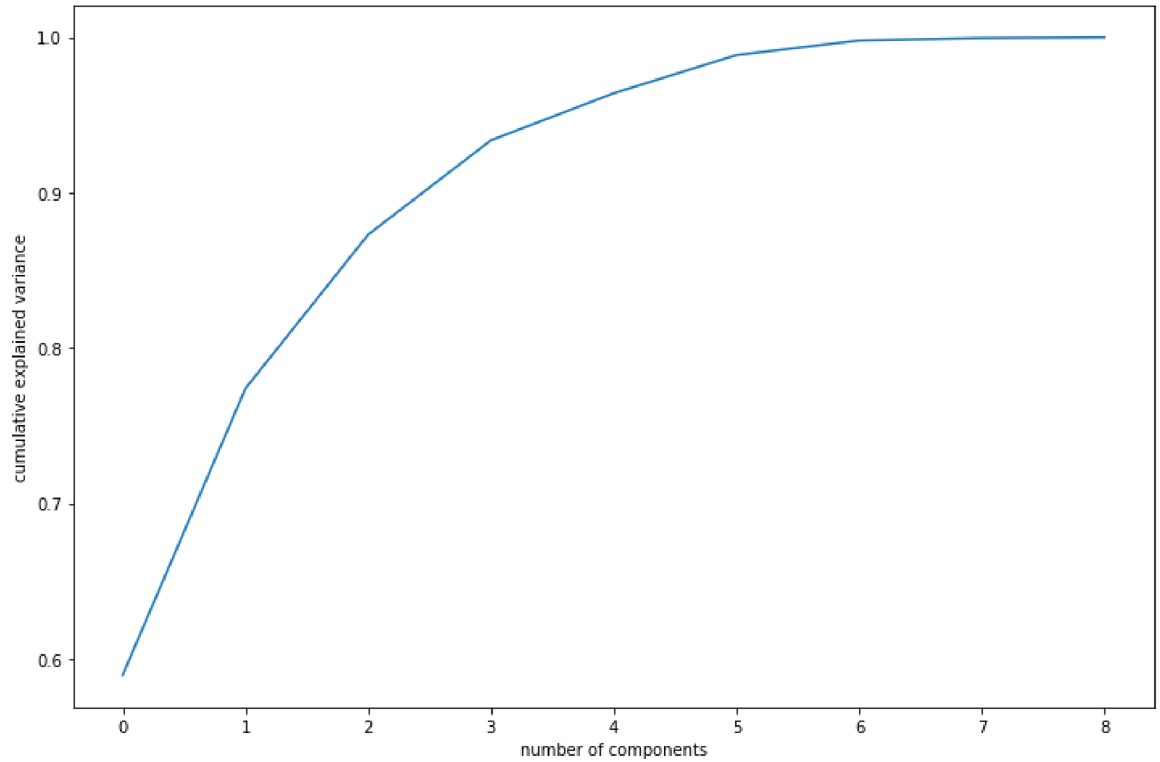
Checking **variance ratio**

```
In [20]: #Let's check the variance ratios
         pca.explained_variance_ratio_

Out[20]: array([5.89372984e-01, 1.84451685e-01, 9.91147170e-02, 6.07227801e-02,
                3.02917253e-02, 2.45982702e-02, 9.39743701e-03, 1.55641971e-03,
                4.93981394e-04])
```

Scree plot

more than 95% of the information is being explained by 5 components



Fitting principal components to the scaled features df

In [22]:
```
#Using incremental PCA for efficiency - saves a lot of time on larger datasets
from sklearn.decomposition import IncrementalPCA
pca_final = IncrementalPCA(n_components=5)
```

In [23]:
```
df_pca = pca_final.fit_transform(df)
df_pca.shape
```

Out[23]:
```
(167, 5)
```

Transposing final df_pca

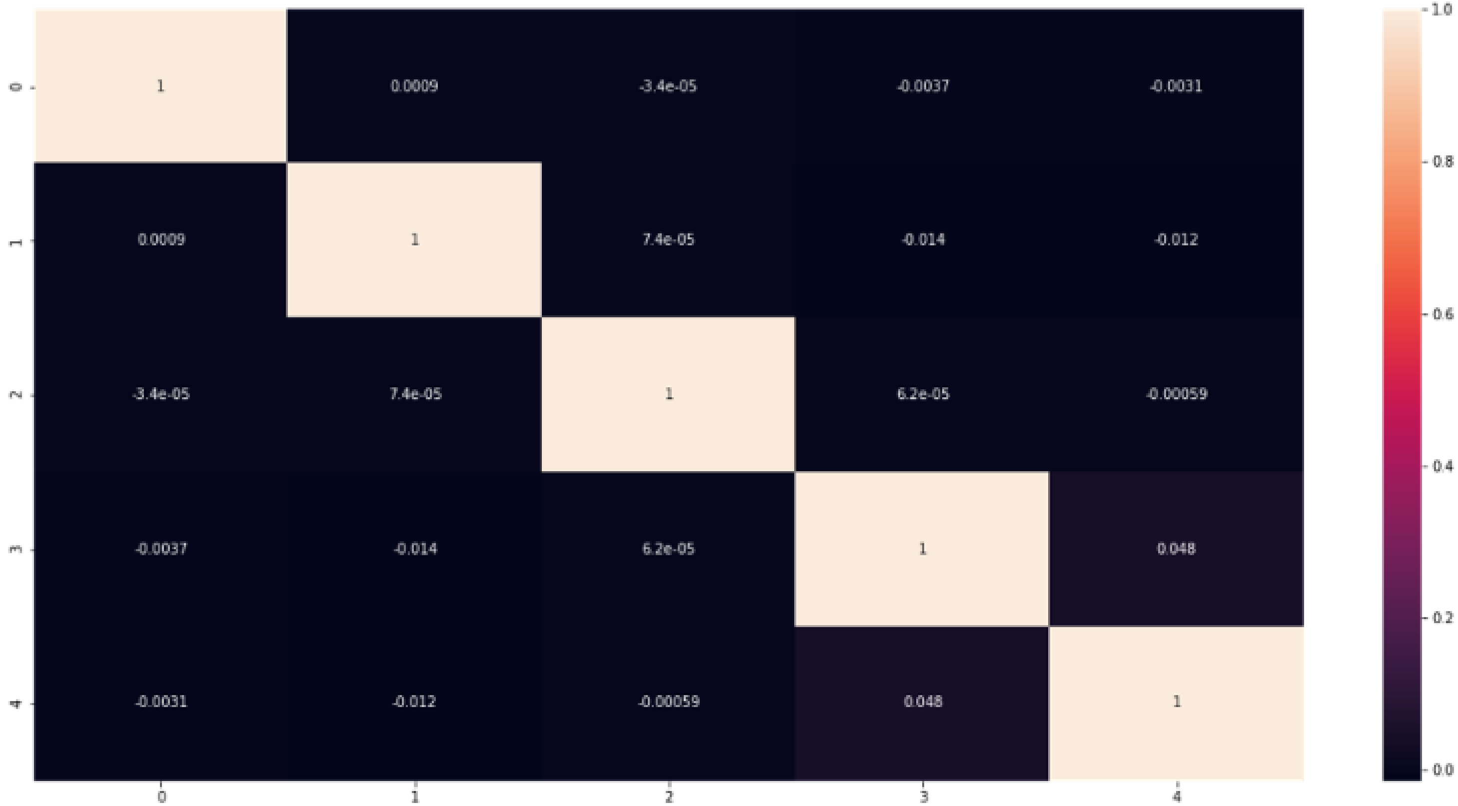
In [25]:
```
#Creating a transpose so that the each column is properly arranged
pc = np.transpose(df_pca)
```

In [26]:
```
pcs_df2 = pd.DataFrame({'PC1':pc[0],'PC2':pc[1]})
pcs_df2.head()
```

Out[26]:

	PC1	PC2
0	-2.628433	1.467845
1	-0.023712	-1.431231
2	-0.457851	-0.677667
3	-2.715305	2.168445
4	0.647157	-1.023327

Checking **correlation** of principal components

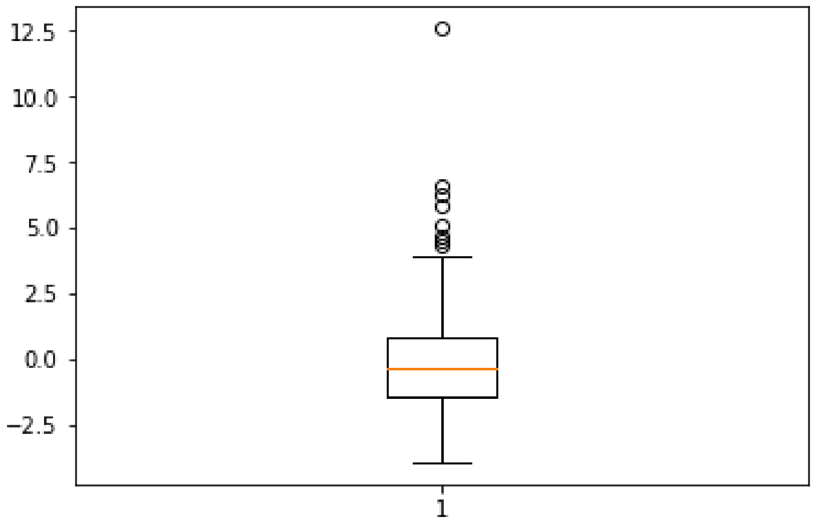



```
In [29]: # 1s -> 0s in diagonals
         corrmat_nodiag = corrmat - np.diagflat(corrmat.diagonal())
         print("max corr:",corrmat_nodiag.max(), ", min corr: ", corrmat_nodiag.min(),)
         # we see that correlations are indeed very close to 0
```

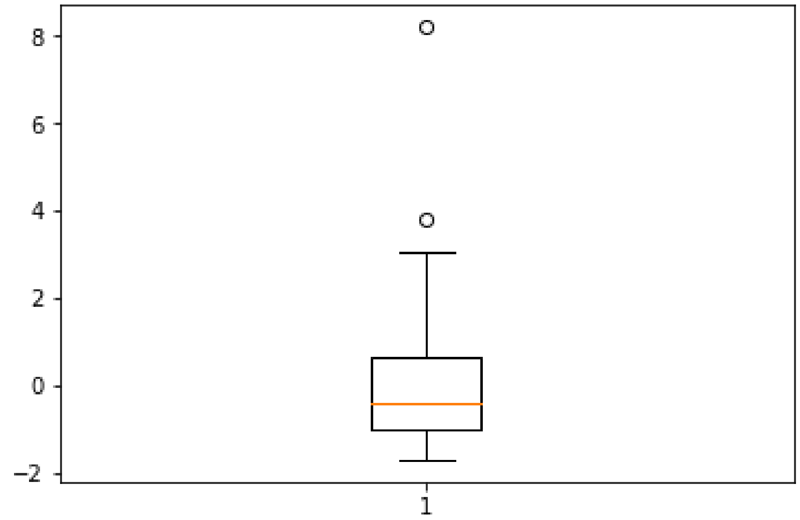
```
max corr: 0.047501009620927404 , min corr:  -0.013777413350897525
```

It seems little or no correlation among the variables

Outlier analysis after PCA



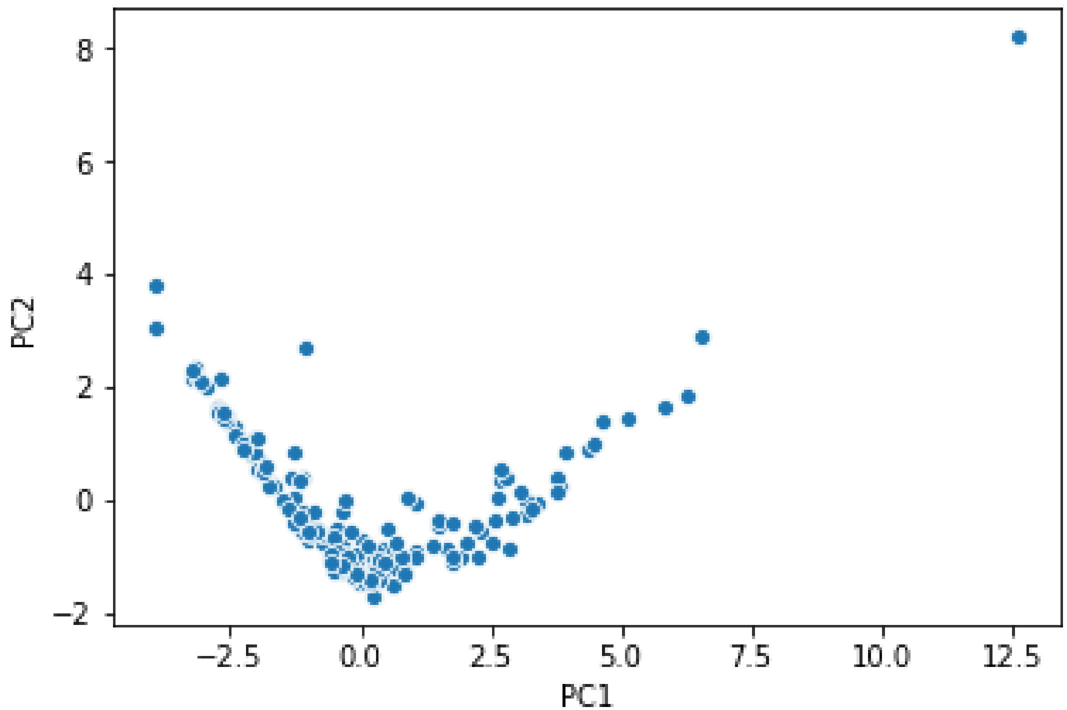
For PC1



For PC2

Since the data is getting lost by treating the outliers therefore we are retaining it and moving forward with it.

Visualizing principal components using scatter plot



Step 5: Hopkins Statistics

```
In [35]: #Let's check the Hopkins measure
         hopkins(pcs_df2)

Out[35]: 0.9659597073422685
```

Since the value is > 0.5 the given dataset has a good tendency to form clusters.

Step 6: K-Means clustering

Importing libraries

```
In [37]: from sklearn.cluster import KMeans
         from sklearn.metrics import silhouette_score
         from scipy.cluster.hierarchy import linkage
         from scipy.cluster.hierarchy import dendrogram
         from scipy.cluster.hierarchy import cut_tree
```

performing k-means with **random clusters** say 4

```
In [38]: # k-means with some arbitrary k
         kmeans = KMeans(n_clusters=4, max_iter=50)
         kmeans.fit(df_2)

Out[38]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=50,
             n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
             random_state=None, tol=0.0001, verbose=0)
```

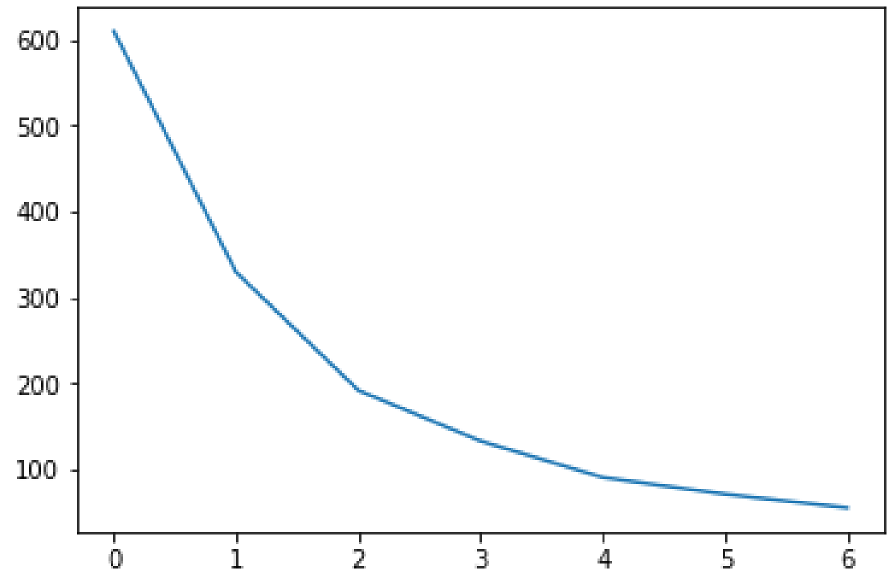
```
In [39]: kmeans.labels_

Out[39]: array([1, 0, 0, 1, 0, 0, 0, 3, 3, 0, 0, 0, 0, 0, 0, 3, 0, 1, 0, 0, 0, 1,
                0, 3, 0, 1, 1, 0, 1, 3, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 3, 0,
                3, 0, 0, 0, 0, 1, 1, 0, 0, 3, 3, 1, 1, 0, 3, 1, 0, 0, 0, 1, 1, 0,
                1, 0, 3, 0, 0, 0, 1, 3, 0, 3, 0, 3, 0, 0, 1, 1, 3, 0, 1, 0, 0, 1,
                1, 0, 0, 2, 0, 1, 1, 0, 0, 1, 3, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
                3, 3, 1, 1, 3, 0, 1, 0, 0, 0, 0, 0, 0, 3, 0, 0, 1, 0, 0, 1, 0, 0,
                1, 3, 0, 3, 1, 1, 0, 3, 0, 0, 1, 0, 3, 3, 0, 1, 0, 1, 1, 0, 0, 0,
                0, 1, 0, 3, 3, 3, 0, 0, 0, 0, 0, 1, 1])
```

Finding optimal number of clusters

Elbow-curve

From the elbow curve, 2 clusters seem to be the optimal number of clusters



Silhouette Analysis

```
For n_clusters=2, the silhouette score is 0.5428022998015879
For n_clusters=3, the silhouette score is 0.5531039376192802
For n_clusters=4, the silhouette score is 0.5608674477746025
For n_clusters=5, the silhouette score is 0.541082010503369
For n_clusters=6, the silhouette score is 0.496751371780202
For n_clusters=7, the silhouette score is 0.4637591769916579
For n_clusters=8, the silhouette score is 0.46484988852400594
```

we are taking 2 clusters into account for final model

Final model with k = 2

In [42]:
```
# final model with k=2
kmeans = KMeans(n_clusters=2, max_iter=50)
kmeans.fit(df_2)
```

Out[42]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=50,
    n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [43]:
```
kmeans.labels_
```

Out[43]:
```
array([1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1])
```

Final data frame of cluster and principal components

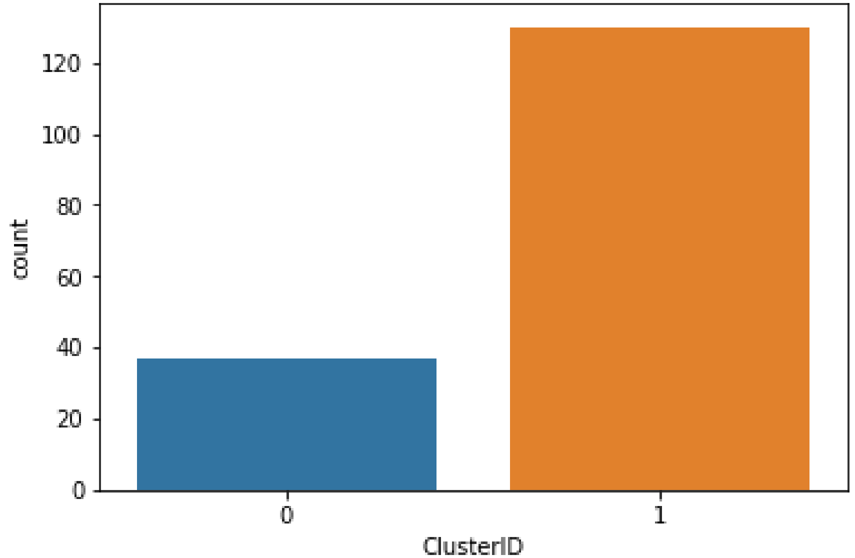
In [44]:
```
# final data frame of clusters
df_3=pcs_df2
df_3.index = pd.RangeIndex(len(df_3.index))
df_km = pd.concat([df_3, pd.Series(kmeans.labels_)], axis=1)
df_km.columns = ['PC1', 'PC2','ClusterID']
df_km.head()
```

Out[44]:

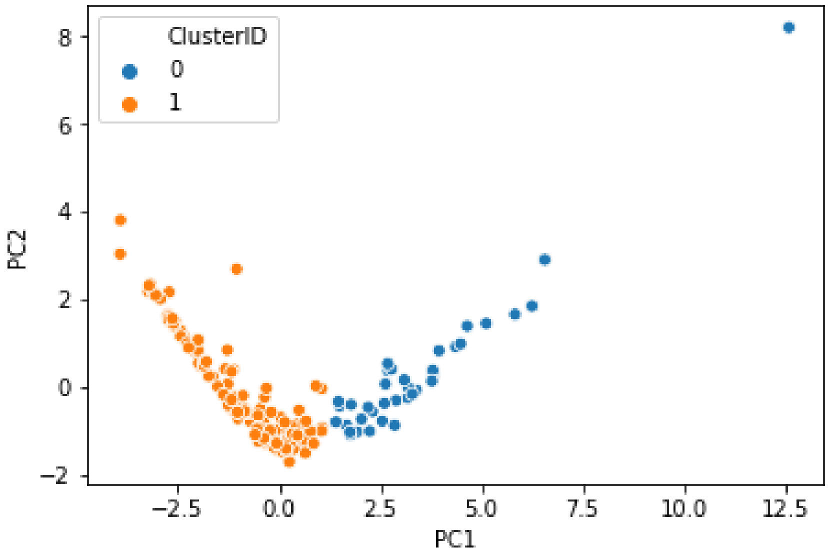
	PC1	PC2	ClusterID
0	-2.628433	1.467845	1
1	-0.023712	-1.431231	1
2	-0.457851	-0.677667	1
3	-2.715305	2.168445	1
4	0.647157	-1.023327	1

Counting the number of clusters

there are 130 cluster 1 and 37 cluster 0



Scatterplot of different clusters



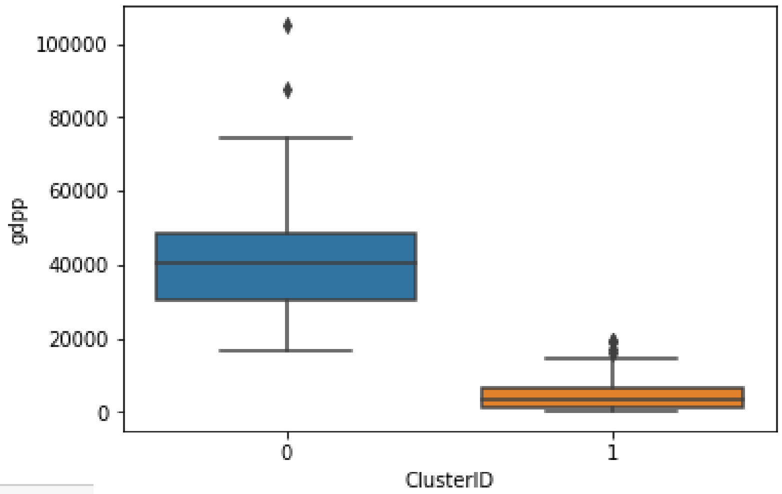
Merging original data set with the final clusters and dropping PC1 and PC2 from the final data set

Out[54]:

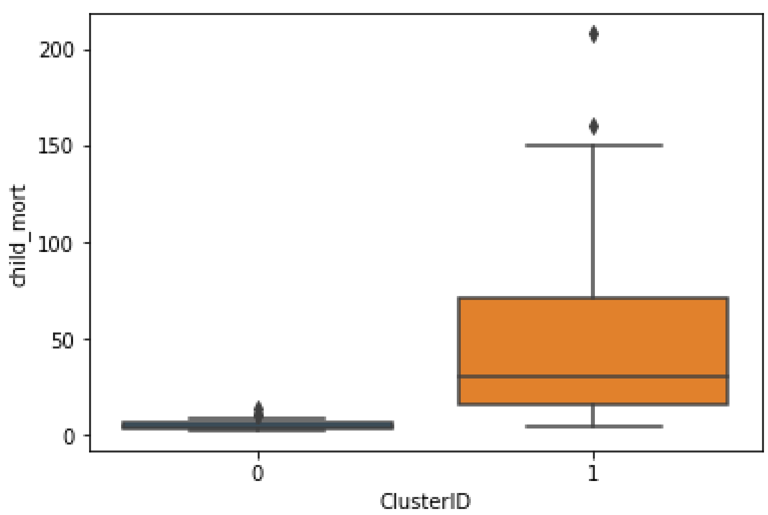
	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	ClusterID
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553	1
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090	1
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460	1
3	Angola	119.0	62.3	2.85	42.9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

Analysing variables cluster wise

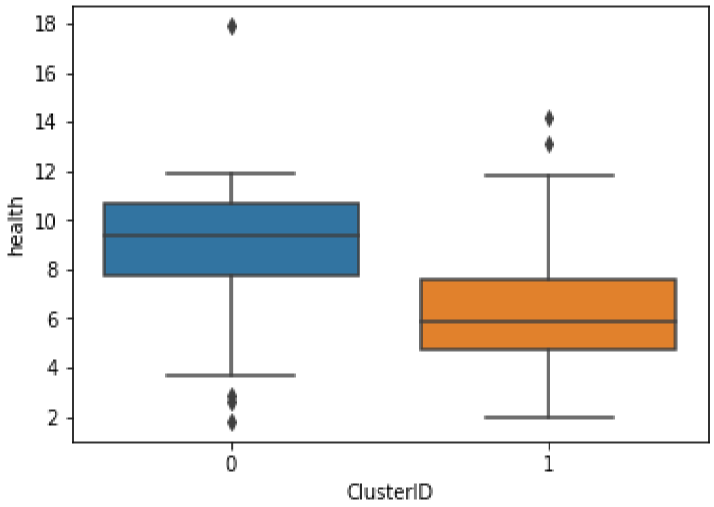
```
# boxplot of gdpp based on ClusterID
sns.boxplot(x = 'ClusterID', y = 'gdpp', data = country)
plt.show()
```



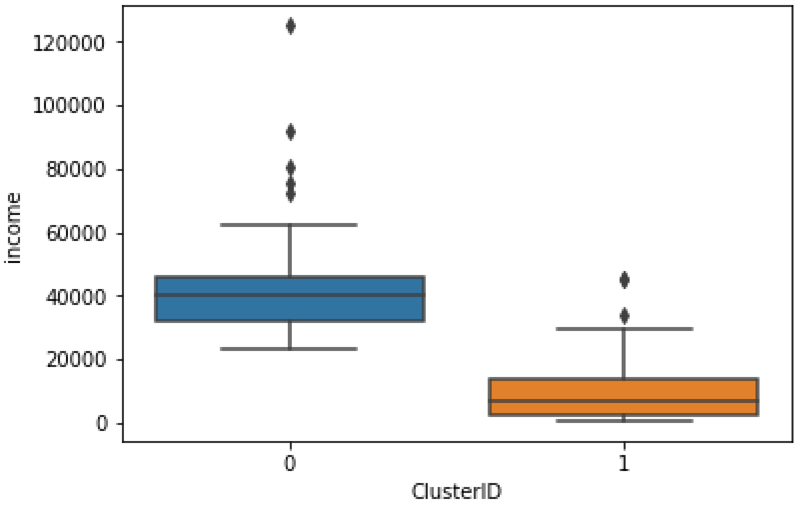
:
```
# boxplot of child_mort based on ClusterID
sns.boxplot(x = 'ClusterID', y = 'child_mort', data = country)
plt.show()
```



```
sns.boxplot(x = 'ClusterID', y = 'health', data = country)
plt.show()
```




```
# boxplot of income based on ClusterID
sns.boxplot(x = 'ClusterID', y = 'income', data = country)
plt.show()
```

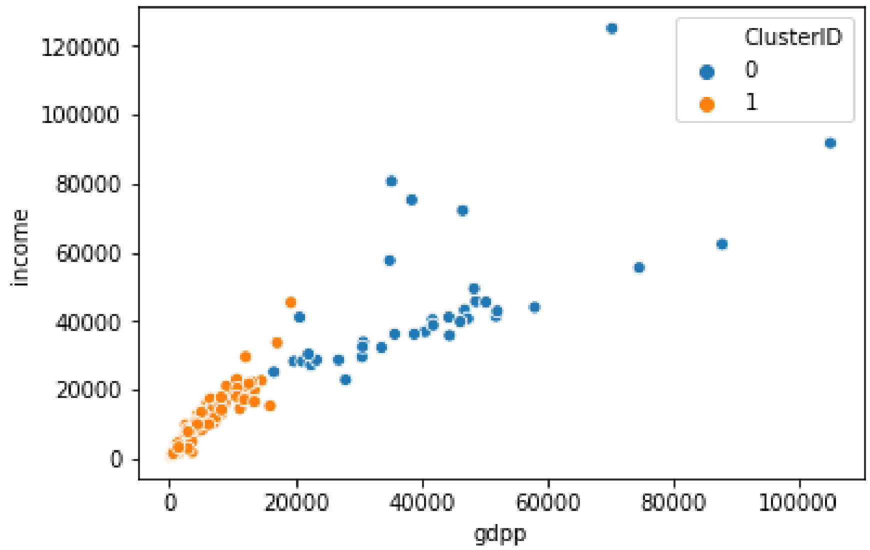


from the above four boxplot we can infer that:

- cluster 1 has low gdpp, high child mortality, low income and lower health than cluster 0
- therefore countries in cluster 1 need more aid than countries in cluster 0

```python
# scatter plot of gdpp and income
sns.scatterplot(x='gdpp',y='income',hue = 'ClusterID', data=country)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fc2a19bc50>
```



As the income increases, gdpp is also increasing
Country with high income and gdpp are mostly present in cluster 0

Step 7: Analyzing the k-means cluster

Grouping variables mean based on ClusterID

```
# grouping different variables and finding the mean based on their ClusterID
clu_gdpp = pd.DataFrame(country.groupby(["ClusterID"]).gdpp.mean())
clu_child_mort = pd.DataFrame(country.groupby(["ClusterID"]).child_mort.mean())
clu_income = pd.DataFrame(country.groupby(["ClusterID"]).income.mean())
clu_health = pd.DataFrame(country.groupby(["ClusterID"]).health.mean())
clu_exports = pd.DataFrame(country.groupby(["ClusterID"]).exports.mean())
clu_imports = pd.DataFrame(country.groupby(["ClusterID"]).imports.mean())
clu_inflation = pd.DataFrame(country.groupby(["ClusterID"]).inflation.mean())
clu_life_expec = pd.DataFrame(country.groupby(["ClusterID"]).life_expec.mean())
clu_total_fer = pd.DataFrame(country.groupby(["ClusterID"]).total_fer.mean())
```

Data frame of the mean of the features cluster-wise

	ClusterID	gdpp	child_mort	income	health	exports	imports	inflation	life_expec	total_fer
0	0	42102.702703	5.237838	45056.756757	8.782973	58.097297	51.281081	2.588432	79.956757	1.755676
1	1	4670.876923	47.671538	9200.484615	6.255769	36.273838	45.640507	9.259954	67.880000	3.287308

Developed countries

```
# developed country
developed_country = country[country['ClusterID'] == 0]
developed_country.head()
```

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	ClusterID
7	Australia	4.8	19.8	8.73	20.9	41400	1.160	82.0	1.93	51900	0
8	Austria	4.3	51.3	11.00	47.8	43200	0.873	80.5	1.44	46900	0
10	Bahamas	13.8	35.0	7.89	43.7	22900	-0.393	73.8	1.86	28000	0
11	Bahrain	8.6	69.5	4.97	50.9	41100	7.440	76.0	2.16	20700	0
15	Belgium	4.5	76.4	10.70	74.7	41100	1.880	80.0	1.86	44400	0

Under developed countries

```
# under developed country
under_developed_country = country[country['ClusterID'] == 1]
under_developed_country.head()
```

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

Binning the countries who need aid from the NGO

```
# binning the country to find the countries who need aid
final=country[country['gdpp']<=4670.88]
final=final[final['child_mort']>= 47.67]
final=final[final['income']<= 9200.48]
```

sorting the data frame to find which country has high child mortality, low health, low income and gdpp, those countries need direct aid from the NGO

```
# sorting the final data frame based on child motality rate, health, income and gdpp
final = final.sort_values(['child_mort', 'health', 'income', 'gdpp'], ascending = [False, True, True, True])
final.head()
```

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	ClusterID
66	Haiti	208.0	15.3	6.91	64.7	1500	5.45	32.1	3.33	662	1
132	Sierra Leone	160.0	16.8	13.10	34.5	1220	17.20	55.0	5.20	399	1
32	Chad	150.0	36.8	4.53	43.5	1930	6.39	56.5	6.59	897	1
31	Central African Republic	149.0	11.8	3.98	26.5	888	2.01	47.5	5.21	446	1
97	Mali	137.0	22.8	4.98	35.1	1870	4.37	59.5	6.55	708	1

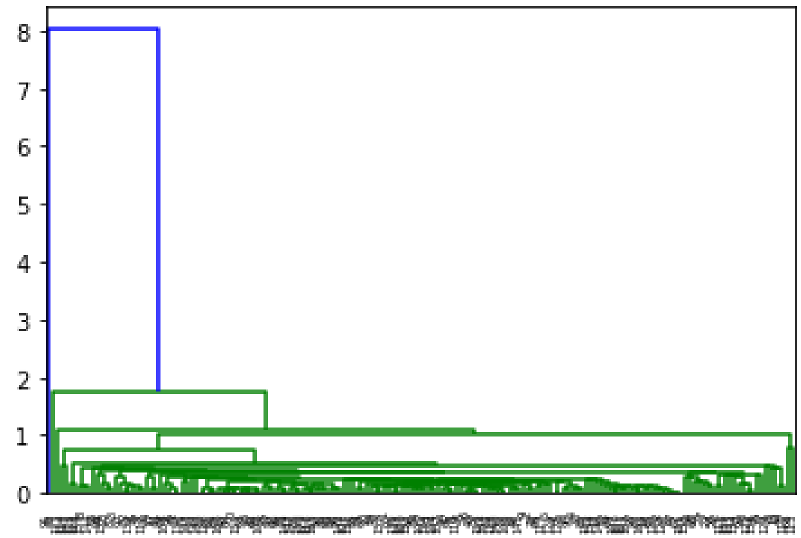
Therefore final list of countries who require aid from the NGO are:

- Haiti
- Sierra Leone
- Chad
- Central African Republic
- Mali

Step 8: Hierarchical Clustering

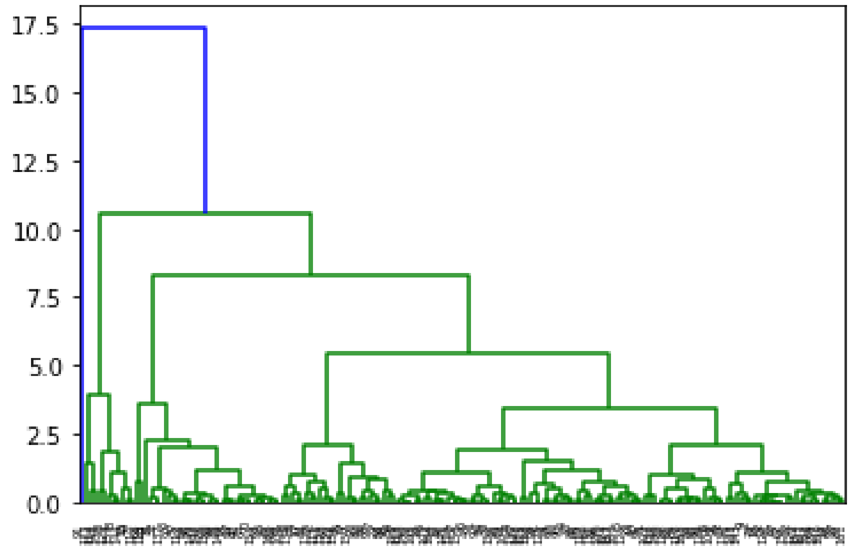
Single linkage

```
# single linkage
mergings = linkage(pcs_df2, method="single", metric='euclidean')
dendrogram(mergings)
plt.show()
```



Complete linkage

```
# complete linkage
mergings = linkage(pcs_df2, method="complete", metric='euclidean')
dendrogram(mergings)
plt.show()
```



from the above dendrogram we can take 3 clusters for our final hierarchical clustering

Dividing features into 3 clusters

```
# 3 clusters
cluster_labels = cut_tree(mergings, n_clusters=3).reshape(-1, )
cluster_labels
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0])
```

```
# assign cluster labels to country dataframe
country['cluster_labels'] = cluster_labels
country.head()
```

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

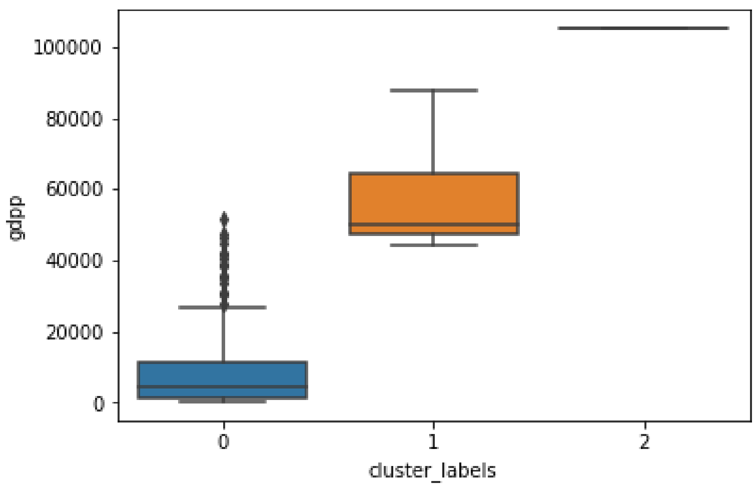
Counting the number of cluster labels

In [108]:
```
# counting the number of cluster_labels
country['cluster_labels'].value_counts()
```

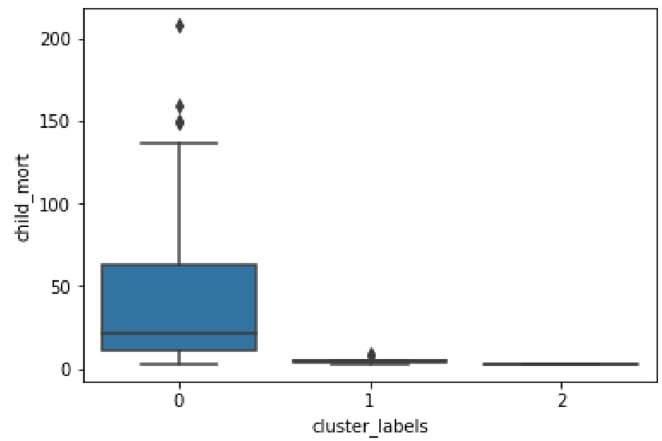
Out[108]:
```
0    155
1     11
2      1
Name: cluster_labels, dtype: int64
```

Analysing variables cluster wise

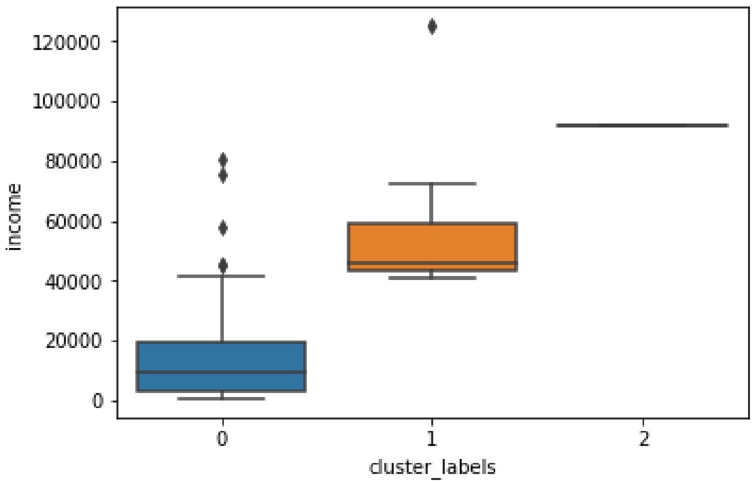
```
# box plot
sns.boxplot(x = 'cluster_labels', y = 'gdpp', data = country)
plt.show()
```



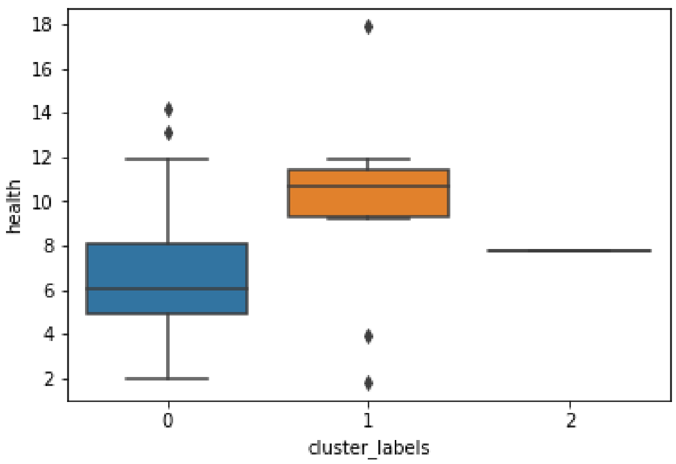

```
# box plot
sns.boxplot(x = 'cluster_labels', y = 'child_mort', data = country)
plt.show()
```



```python
# box plot
sns.boxplot(x = 'cluster_labels', y = 'income', data = country)
plt.show()
```



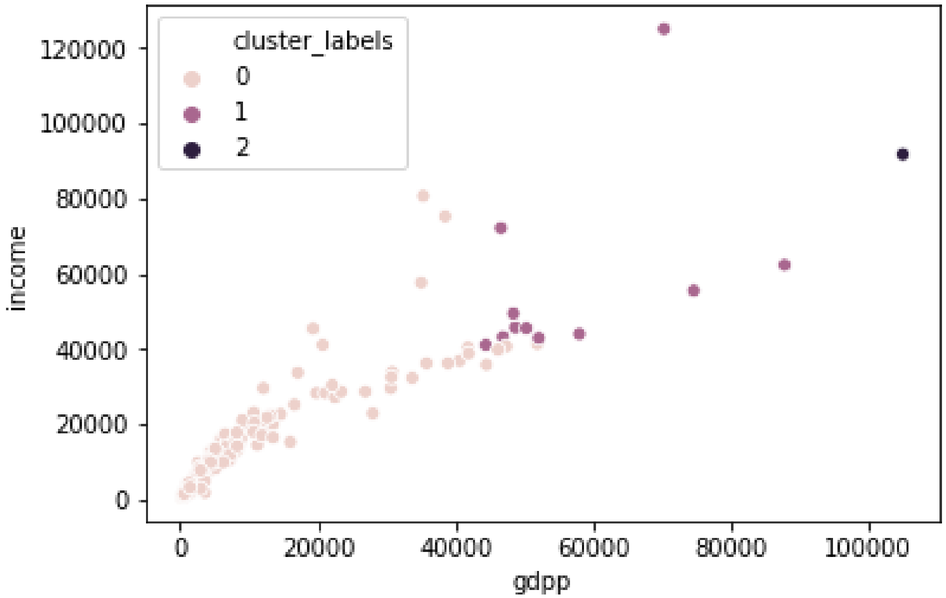

```python
# boxplot
sns.boxplot(x = 'cluster_labels', y = 'health', data = country)
plt.show()
```



from the above boxplot we can see that:

- cluster 0 has low gdpp, high child_mort, low income and low health than the rest of the clusters
- therefore countries in cluster 0 need more aid than the rest

Cluster 0 falls under low income and low gdpp



Step 9: Analyzing the hierarchical clusters

Grouping variables mean based on cluster_labels

```
# grouping different variables and finding the mean based on their cluster_labels
clu_gdpp = pd.DataFrame(country.groupby(["cluster_labels"]).gdpp.mean())
clu_child_mort = pd.DataFrame(country.groupby(["cluster_labels"]).child_mort.mean())
clu_income = pd.DataFrame(country.groupby(["cluster_labels"]).income.mean())
clu_health = pd.DataFrame(country.groupby(["cluster_labels"]).health.mean())
clu_exports = pd.DataFrame(country.groupby(["cluster_labels"]).exports.mean())
clu_imports = pd.DataFrame(country.groupby(["cluster_labels"]).imports.mean())
clu_inflation = pd.DataFrame(country.groupby(["cluster_labels"]).inflation.mean())
clu_life_expec = pd.DataFrame(country.groupby(["cluster_labels"]).life_expec.mean())
clu_total_fer = pd.DataFrame(country.groupby(["cluster_labels"]).total_fer.mean())
```

Data frame of the mean of the features cluster-wise

	cluster_labels	gdpp	child_mort	income	health	exports	imports	inflation	life_expec	total_fer
0	0	9238.154839	40.883226	13837.180645	6.593419	38.144510	45.395909	8.238277	69.772903	3.039161
1	1	57100.000000	4.672727	56972.727273	9.860909	70.709091	59.300000	1.728455	80.609091	1.782727
2	2	105000.000000	2.800000	91700.000000	7.770000	175.000000	142.000000	3.620000	81.300000	1.630000

Binning the countries who need aid from the NGO

```
# binning the country to find the countries who need aid
final=country[country['gdpp']<=9238.15]
final=final[final['child_mort']>= 40.88]
final=final[final['income']<= 13837.18]
```

sorting the data frame to find which country has high child mortality, low health, low income and gdpp, those countries need direct aid from the NGO

```
# sorting the final data frame based on child motality rate, health, income and gdpp
final = final.sort_values(['child_mort', 'health', 'income', 'gdpp'], ascending = [False, True, True, True])
final.head()
```

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	ClusterID	cluster_labels
66	Haiti	208.0	15.3	6.91	64.7	1500	5.45	32.1	3.33	662	1	0
132	Sierra Leone	160.0	16.8	13.10	34.5	1220	17.20	55.0	5.20	399	1	0
32	Chad	150.0	36.8	4.53	43.5	1930	6.39	56.5	6.59	897	1	0
31	Central African Republic	149.0	11.8	3.98	26.5	888	2.01	47.5	5.21	446	1	0
97	Mali	137.0	22.8	4.98	35.1	1870	4.37	59.5	6.55	708	1	0

Therefore final list of countries who require aid from the NGO are:

- Haiti
- Sierra Leone
- Chad
- Central African Republic
- Mali

which is same as k-means clustering