CLUSTERING FOR

HELP INTERNATIONAL NGO

01.03.2021

ADVAIT VATS

PROBLEM STATEMENT

- Due to the data not including any labels we will perform unsupervised learning using clusters.
- The aim of the analysis is to identify, out of 167 countries, which countries are in the direst need of aid from the NGO.
- To verify our result we will use both hierarchical and kmean clustering to find a cluster which will only include the countries in the direst need to allocate funds.
- Three variables are to be used— GDPP, income, child mortality rate— for analysing and shortlisting the countries for aid.

DATA PREPARATION

UNDERSTANDING THE DATASET

The dataset has 10 columns where the export, health, and import are given in percentage of gdpp. I used the countries column instead of the index to allow ease of calculation and analysis.

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

INSPECTING THE DATATYPE AND CHECKING FOR NULL VALUES

Describe method was used to check for any null values that might be present in the dataset. While also checking the datatype of each column to view how to commute them and choose the method of analysis.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   country     167 non-null    object
 1   child_mort  167 non-null    float64
 2   exports     167 non-null    float64
 3   health      167 non-null    float64
 4   imports     167 non-null    float64
 5   income      167 non-null    int64
 6   inflation   167 non-null    float64
 7   life_expec  167 non-null    float64
 8   total_fer   167 non-null    float64
 9   gdpp        167 non-null    int64
dtypes: float64(7), int64(2), object(1)
memory usage: 13.2+ KB
```

SCALING OF THE DATA

Scaler method was used to standardise the dataset. This is an important step as we can notice the units for most columns vary greatly and therefore could create a model which was not optimal for finding the solution or biased towards a column.

```
#scaling and dropping the categorical variable
ss = StandardScaler()
countries_data_sca = ss.fit_transform(countries_data)
countries_data_sca
```

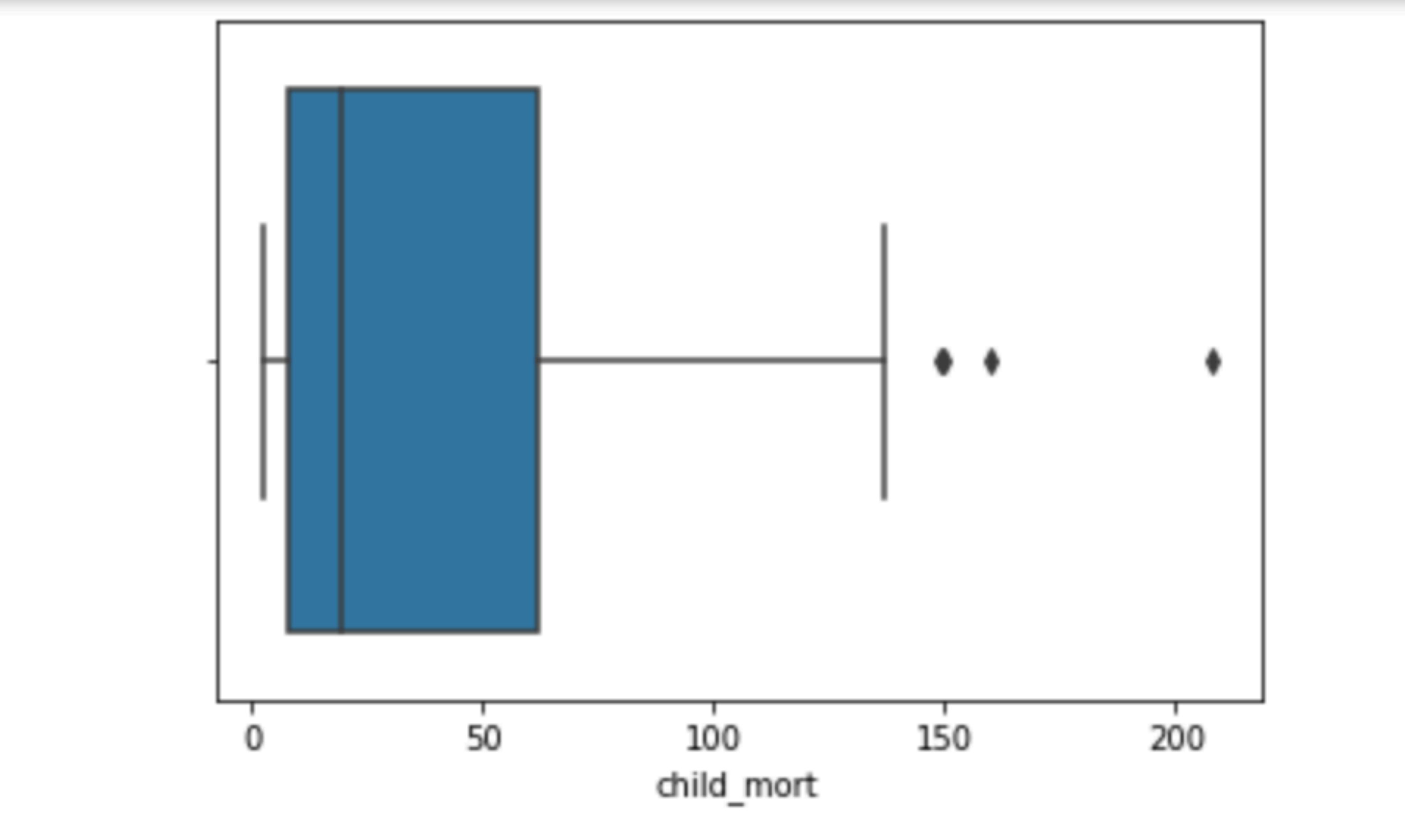
```
array([[ 1.29153238, -0.56962212, -0.56746969, ..., -1.61909203,
         1.90288227, -0.70225949],
       [-0.5389489 , -0.47385792, -0.44070503, ...,  0.64786643,
        -0.85997281, -0.49872564],
       [-0.27283273, -0.42399973, -0.48665504, ...,  0.67042323,
        -0.0384044 , -0.47743428],
       ...,
       [-0.37231541, -0.49160668, -0.54071936, ...,  0.28695762,
        -0.66120626, -0.65869853],
       [ 0.44841668, -0.53995007, -0.55291802, ..., -0.34463279,
         1.14094382, -0.65869853],
       [ 1.11495062, -0.52701632, -0.54274443, ..., -2.09278484,
         1.6246091 , -0.6500669 ]])
```

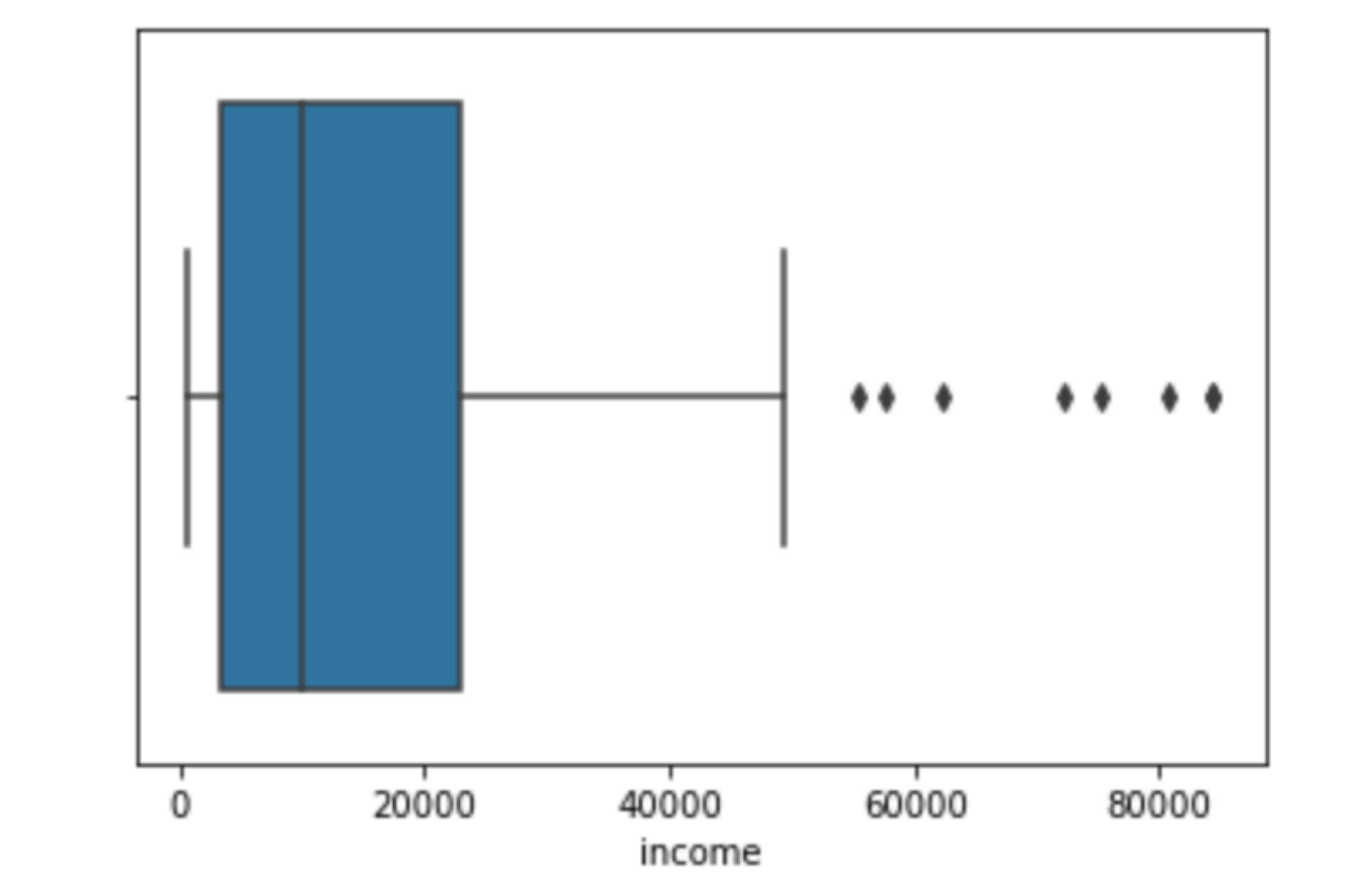
```
countries_data_sca = pd.DataFrame(countries_data_sca)
countries_data_sca.columns = col
countries_data_sca.head()
```

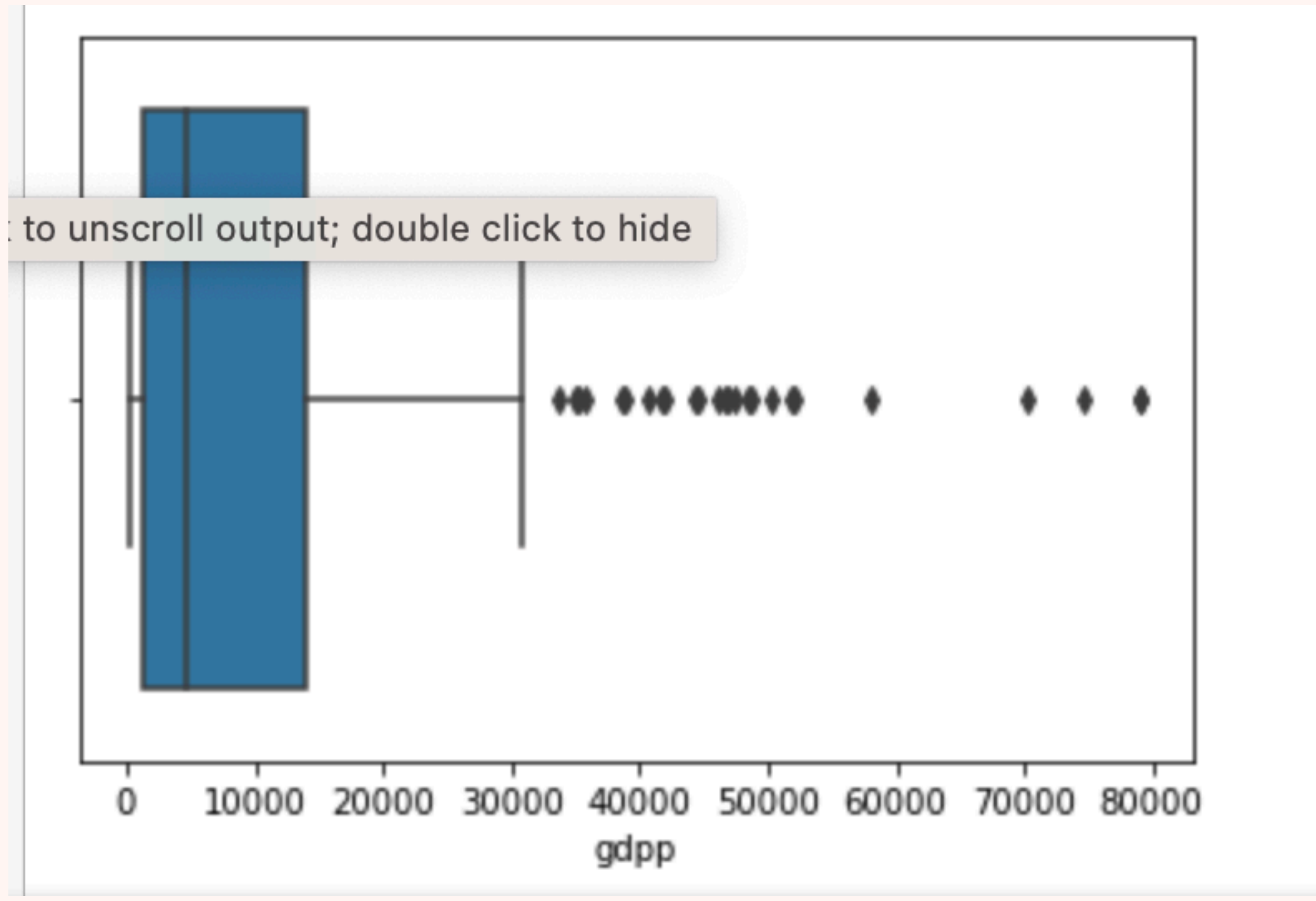
	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	1.291532	-0.569622	-0.567470	-0.432276	-0.851668	0.157336	-1.619092	1.902882	-0.702259
1	-0.538949	-0.473858	-0.440705	-0.313677	-0.386946	-0.312347	0.647866	-0.859973	-0.498726
2	-0.272833	-0.424000	-0.486655	-0.353720	-0.221053	0.789274	0.670423	-0.038404	-0.477434
3	2.007808	-0.381249	-0.534548	-0.345953	-0.612045	1.387054	-1.179234	2.128151	-0.530950
4	-0.695634	-0.086742	-0.178307	0.040735	0.125254	-0.601749	0.704258	-0.541946	-0.032042

OUTLIER TREATMENT

Since we need the one in direst need we will overlook the ones which are below the first quartile. Whereas for the higher value we used soft-capping.



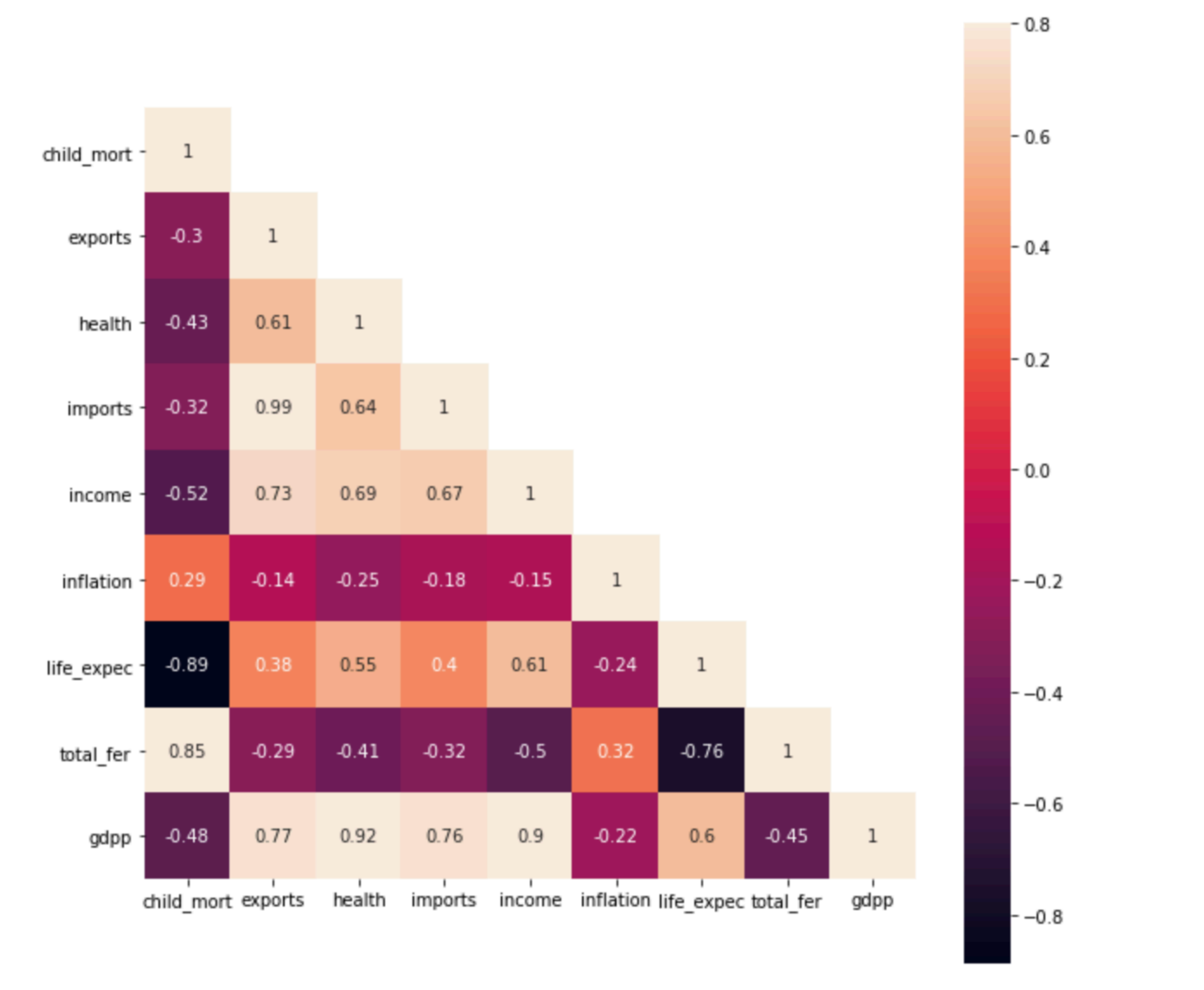




DATA VISUALIZATION

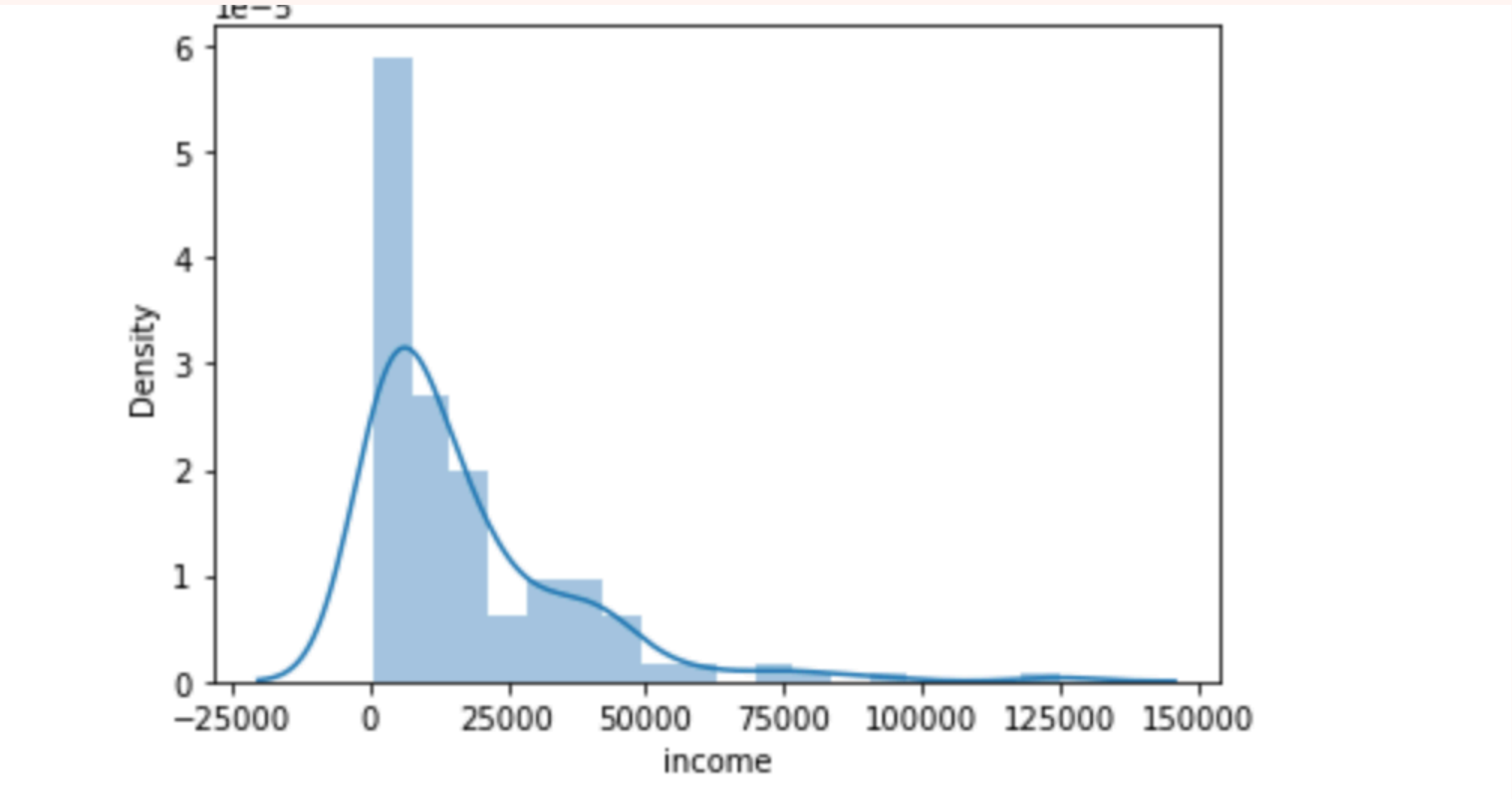
HEAT MAP CORRELATION ANALYSIS

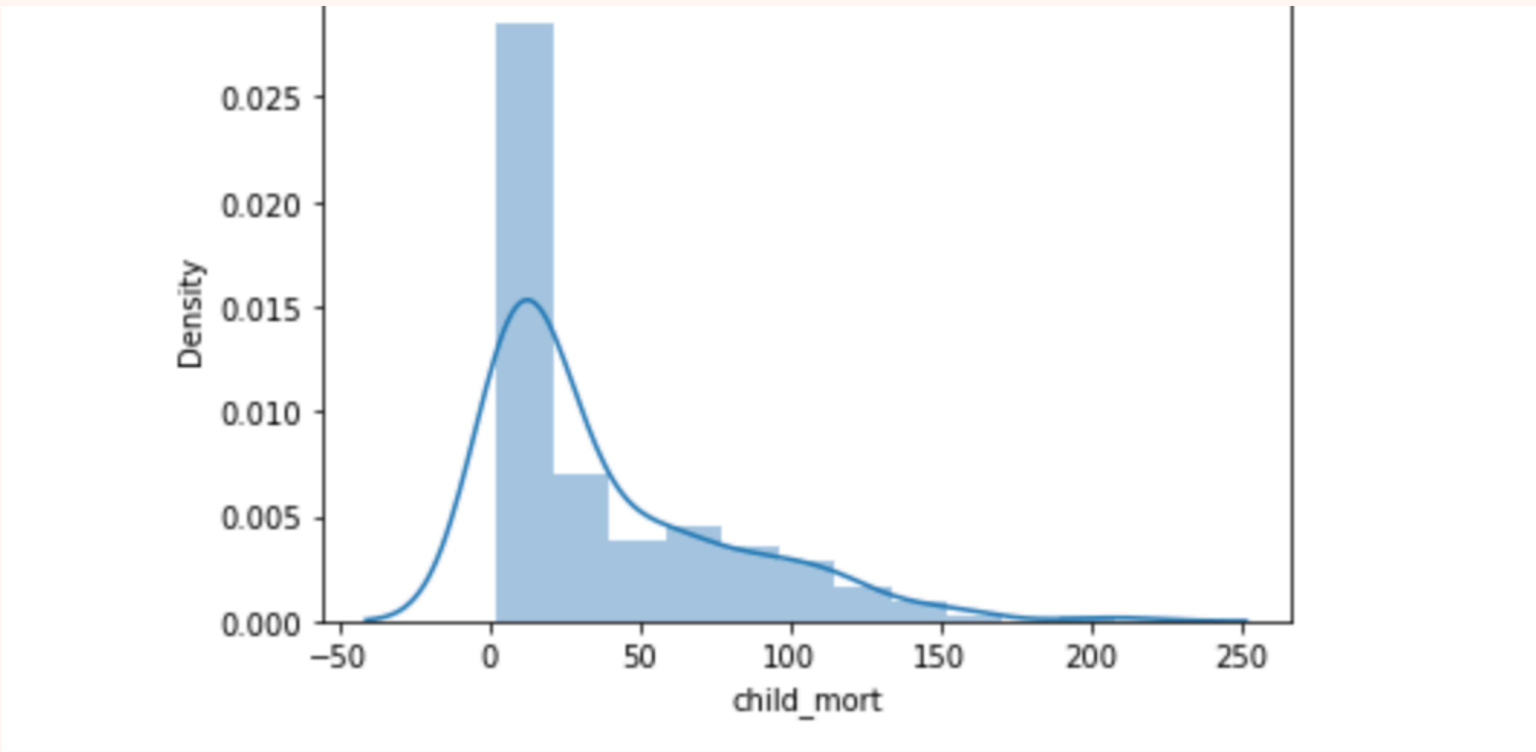
The heat map correlation analysis was used to check for correlation between different elements and how each element is affected by another column. GDPP and child mortality was observed to be negatively correlated which made sense. While health, income and import-export showed a strong positive correlation with the gdpp. As expected, life expectancy decreased with increase in fertility.

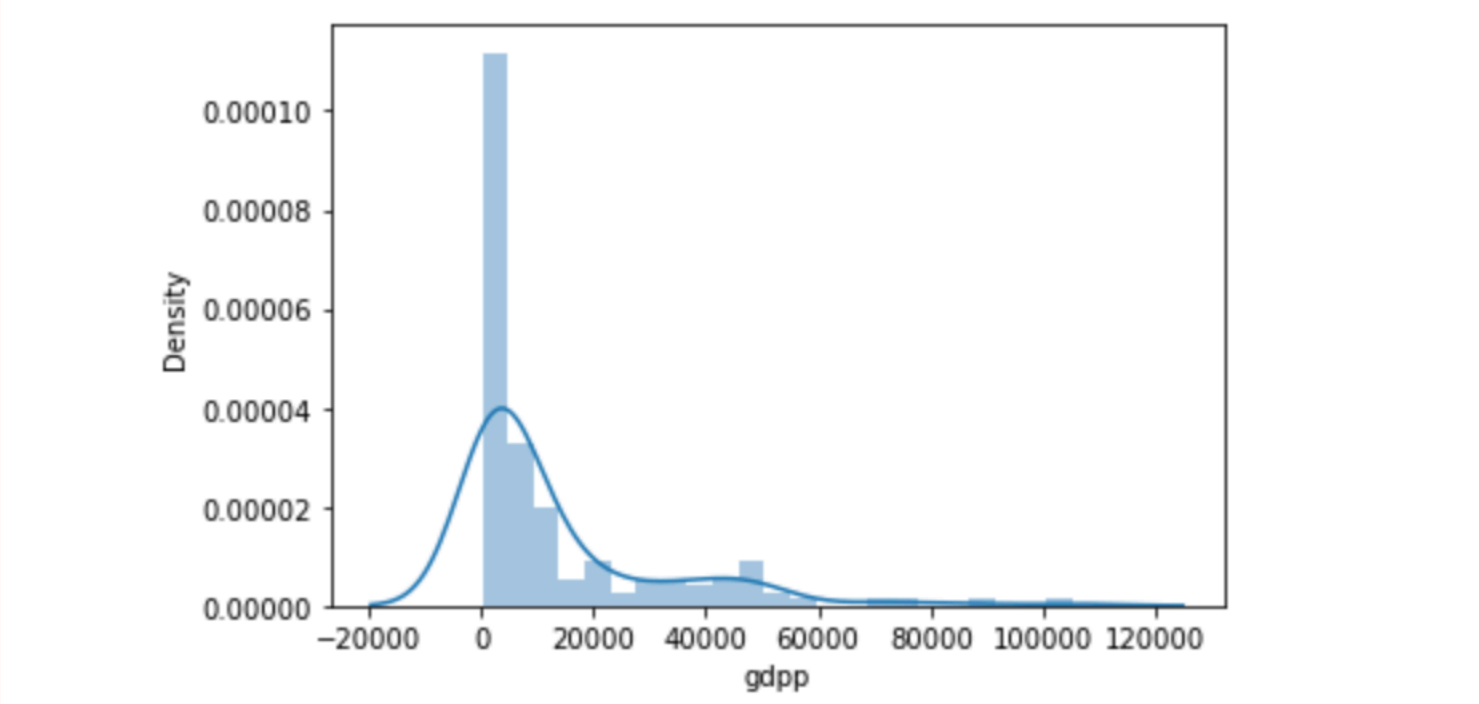


DISTRIBUTION OF THE COLUMNS

The distribution of the columns as it can be noticed is normally distributed. As it is observed the mean is at 0. The GDPP and both the column shows a local maxima again after the global maxima.







CHOOSING THE BEST VALUE FOR K

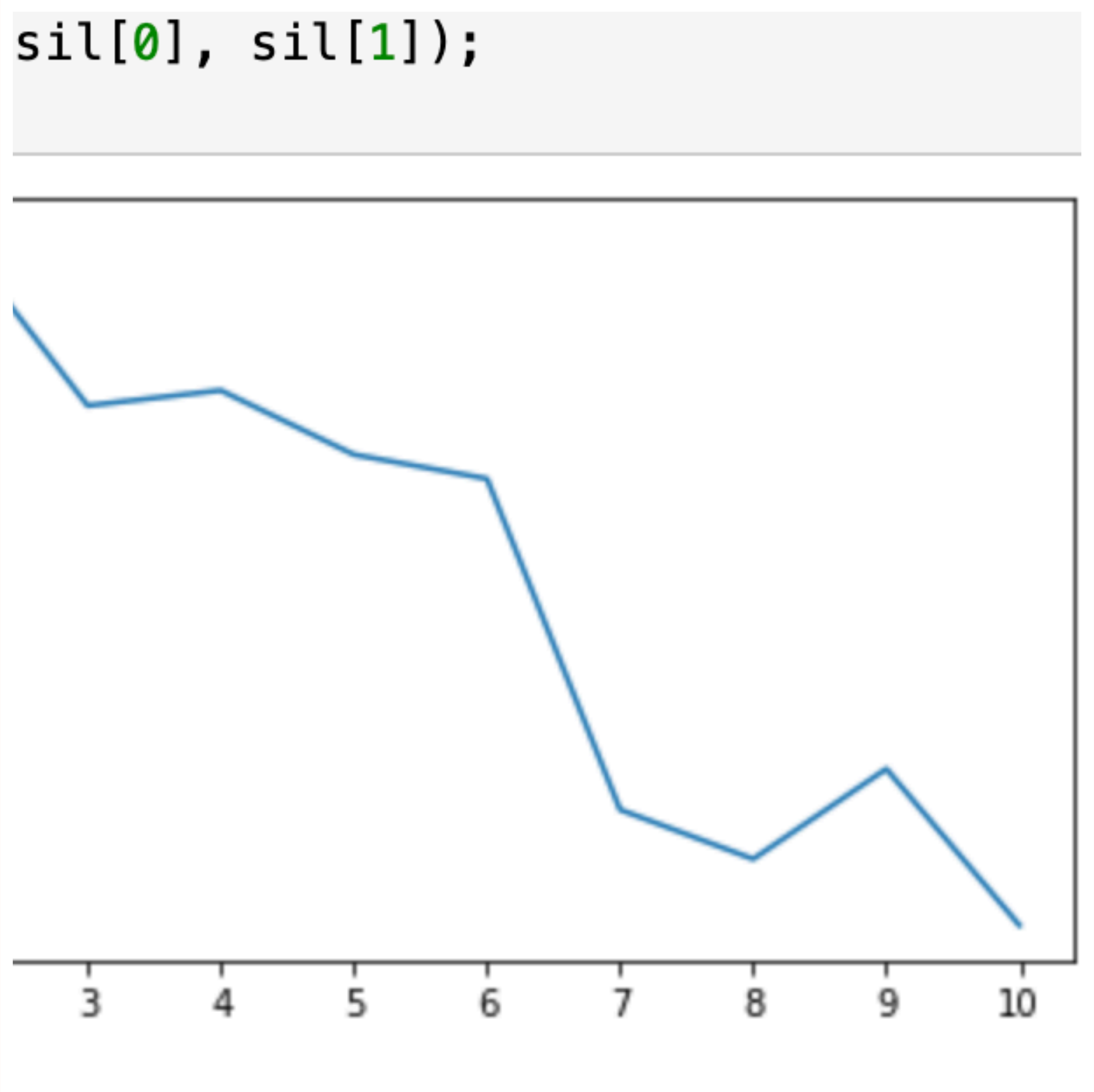
HOPKINS STATISTIC

- **This was used to check if the data was indeed random or was identical to other form of distribution**

```
#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, return_distanc
        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
```

SILHOUETTE SCORE

Silhouette score was used to find best possible value for the number of clusters. Using this method we notice the 4 has the highest peak after 2. Therefore, we choose k=4.



SSD ELBOW

We used another method accompanied to silhouette score to make sure the value of k was optimal. Using this we realised that k=5 was also optimal. So, we chose k=4,5. For clustering.

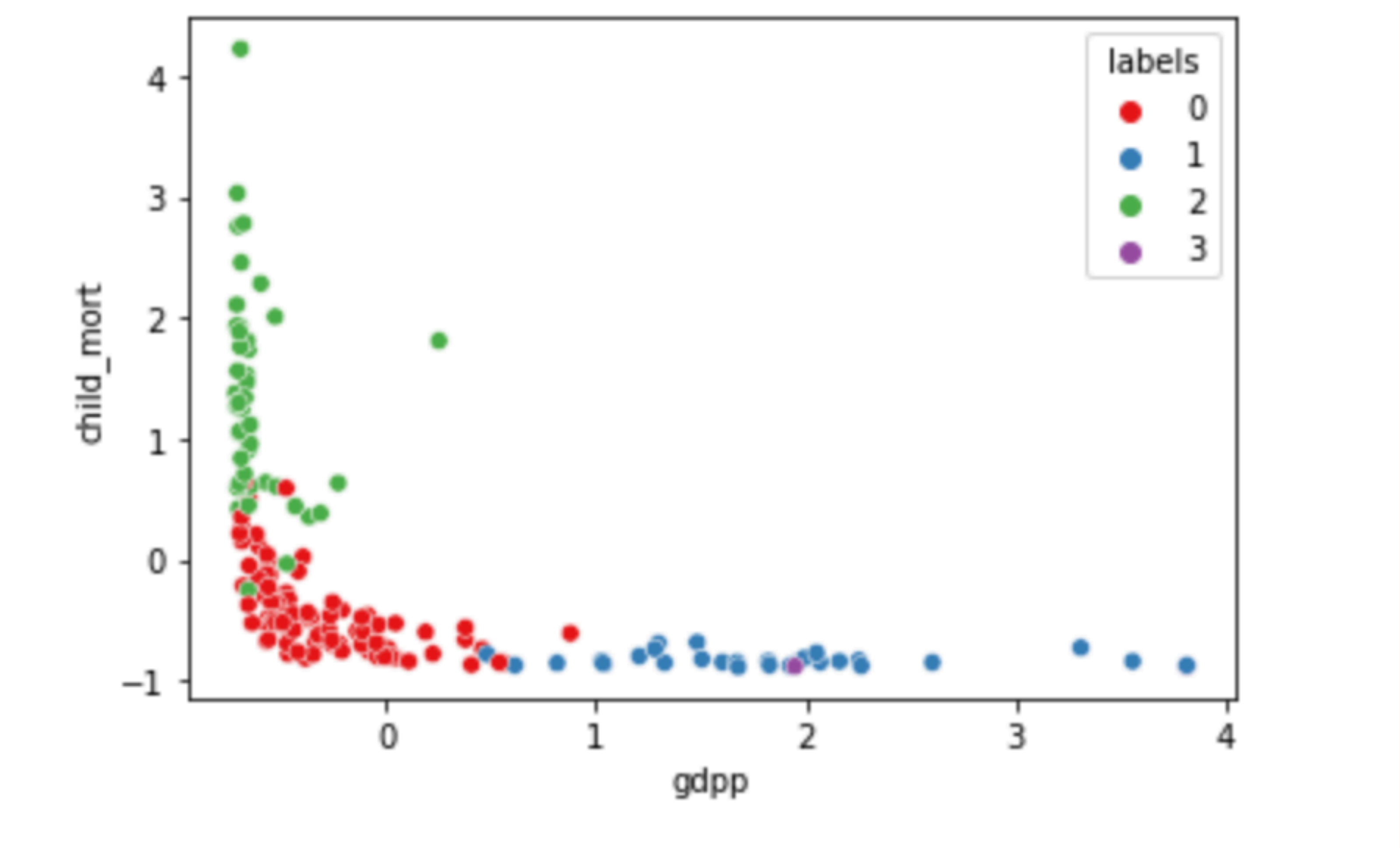
```
# SSD: Elbow

ssd = []
for k in range(2,11):
    kmean = KMeans(n_clusters=k).fit(countries_data_sca)
    ssd.append([k, kmean.inertia_])
ssd
```

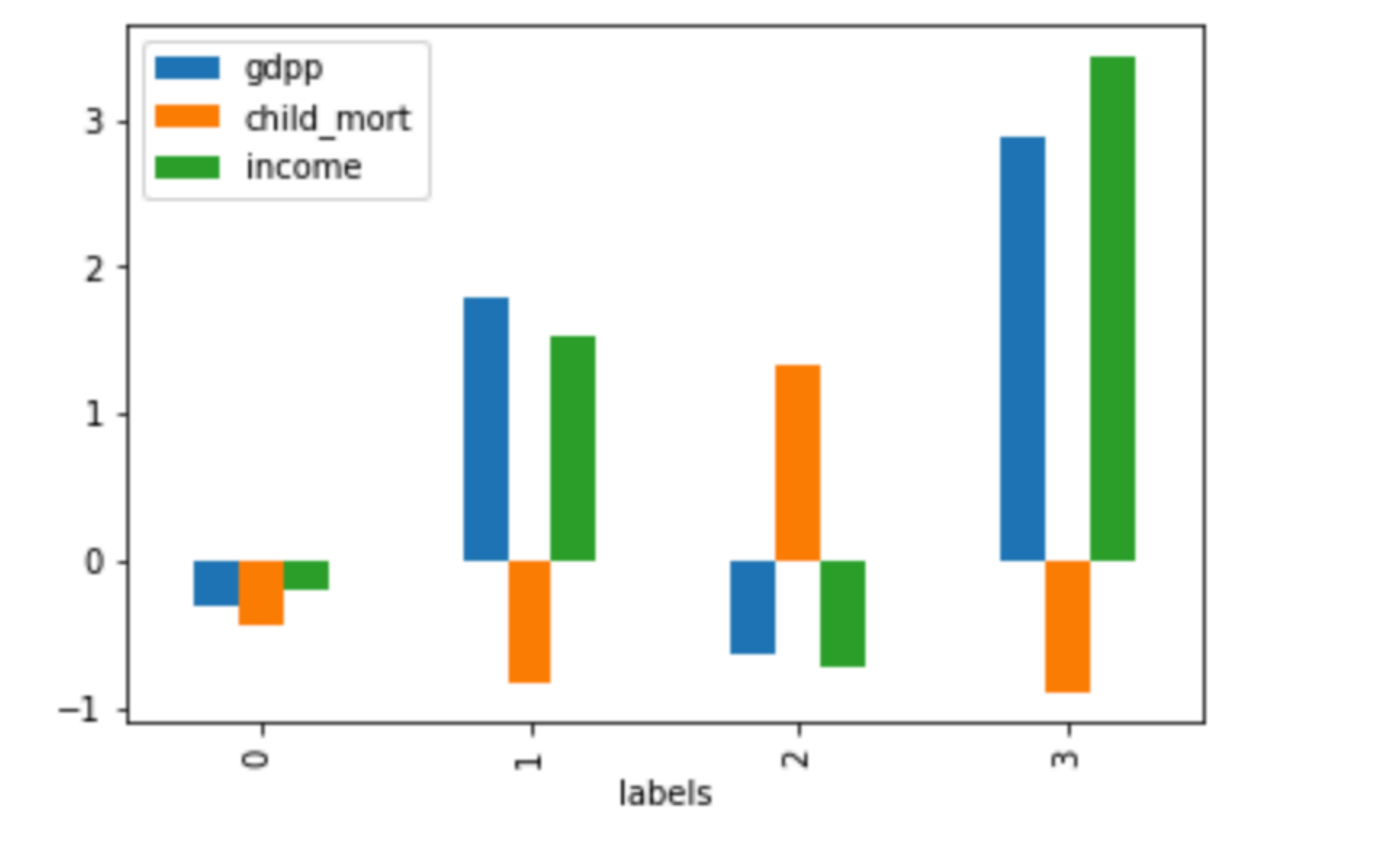
```
[[2, 890.3955365935128],
 [3, 599.6079104407684],
 [4, 483.6829624932816],
 [5, 408.6453223160747],
 [6, 342.3954202449514],
 [7, 305.31563351501035],
 [8, 273.8254836048561],
 [9, 247.5941881253085],
 [10, 221.29293102678832]]
```

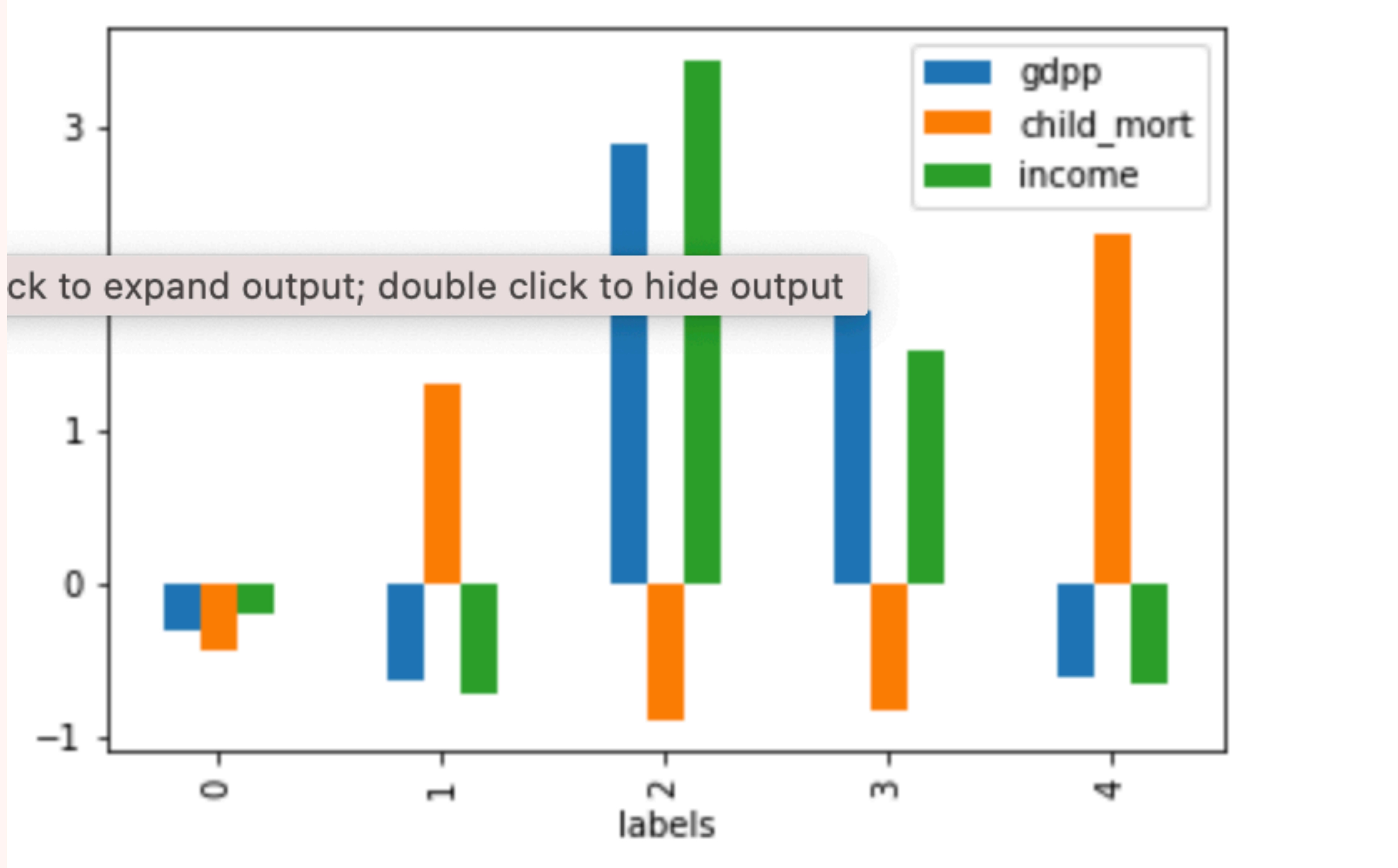
CLUSTERING USING KMEAN

KMEAN; K=4,5

Using kmean various bins were formed. As we can see label 2 and label 1 when k=5— has the highest child mortality rate, and also the most income and GDPP.

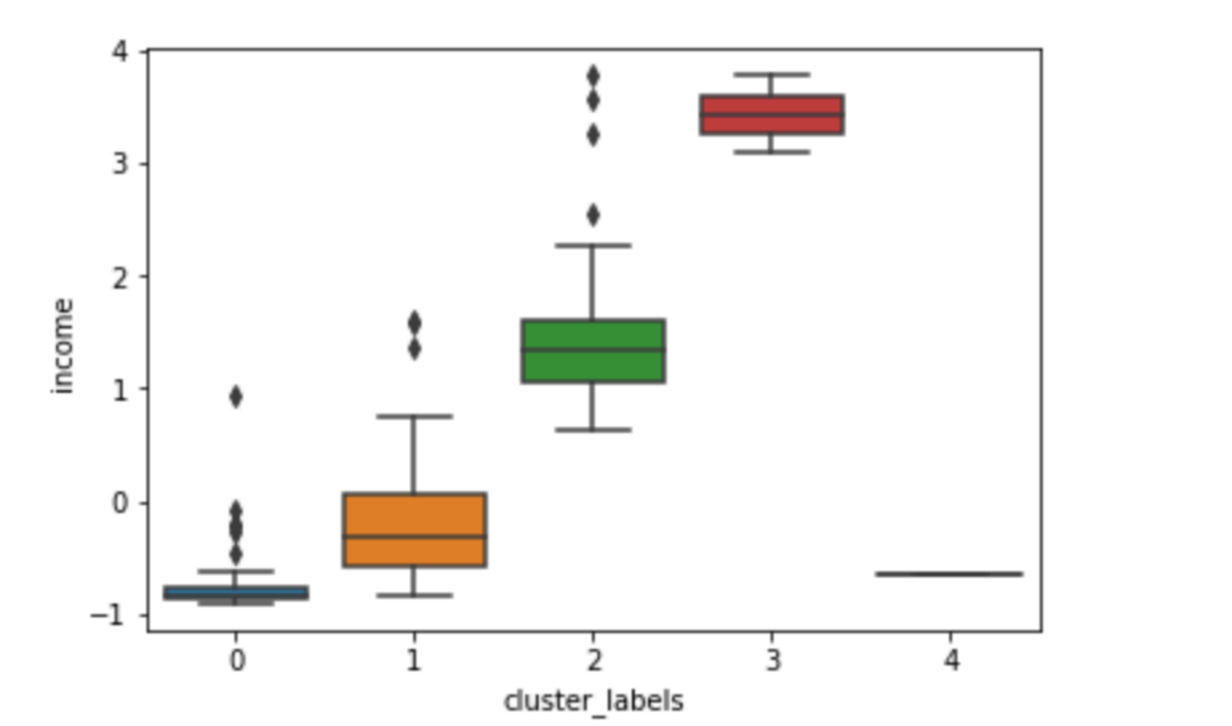


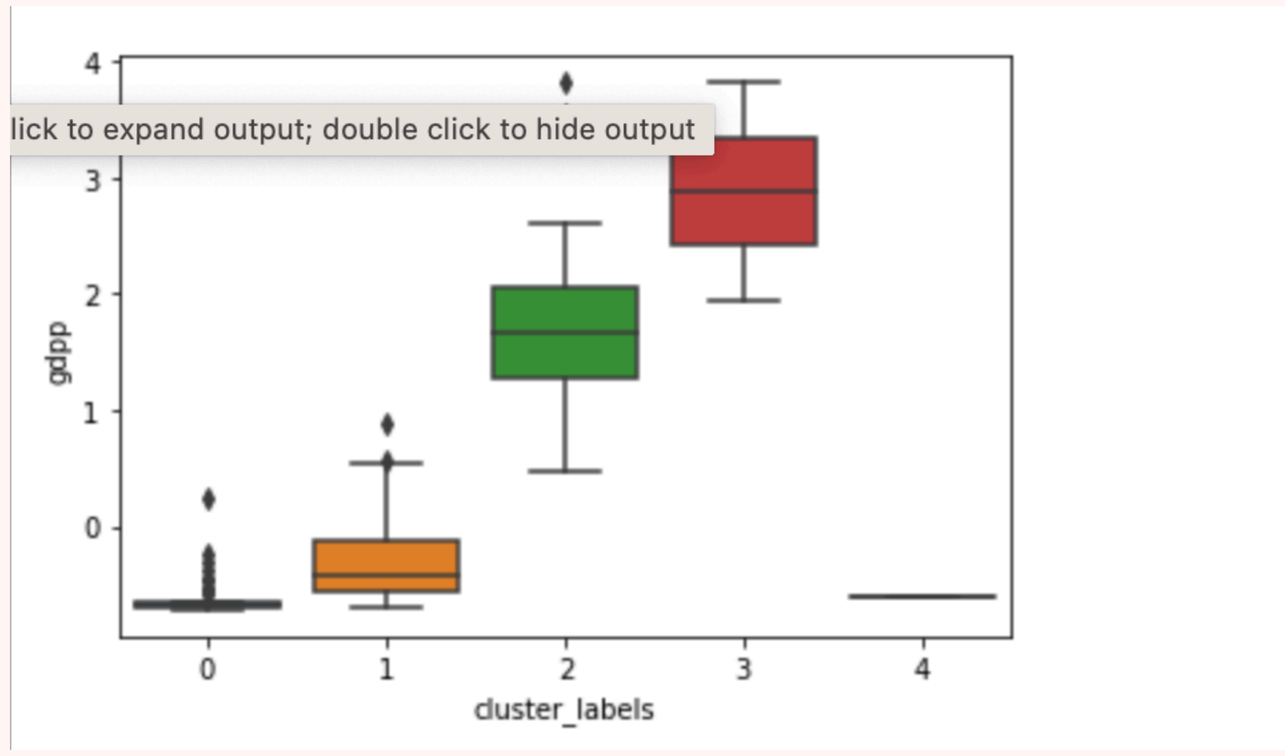


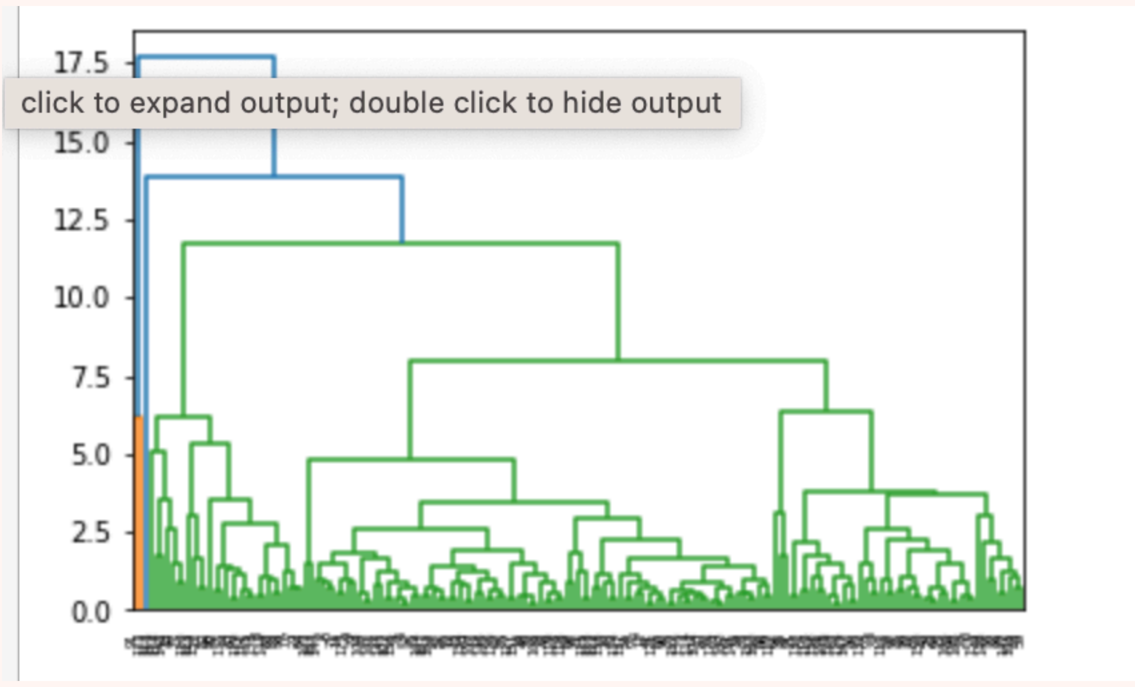


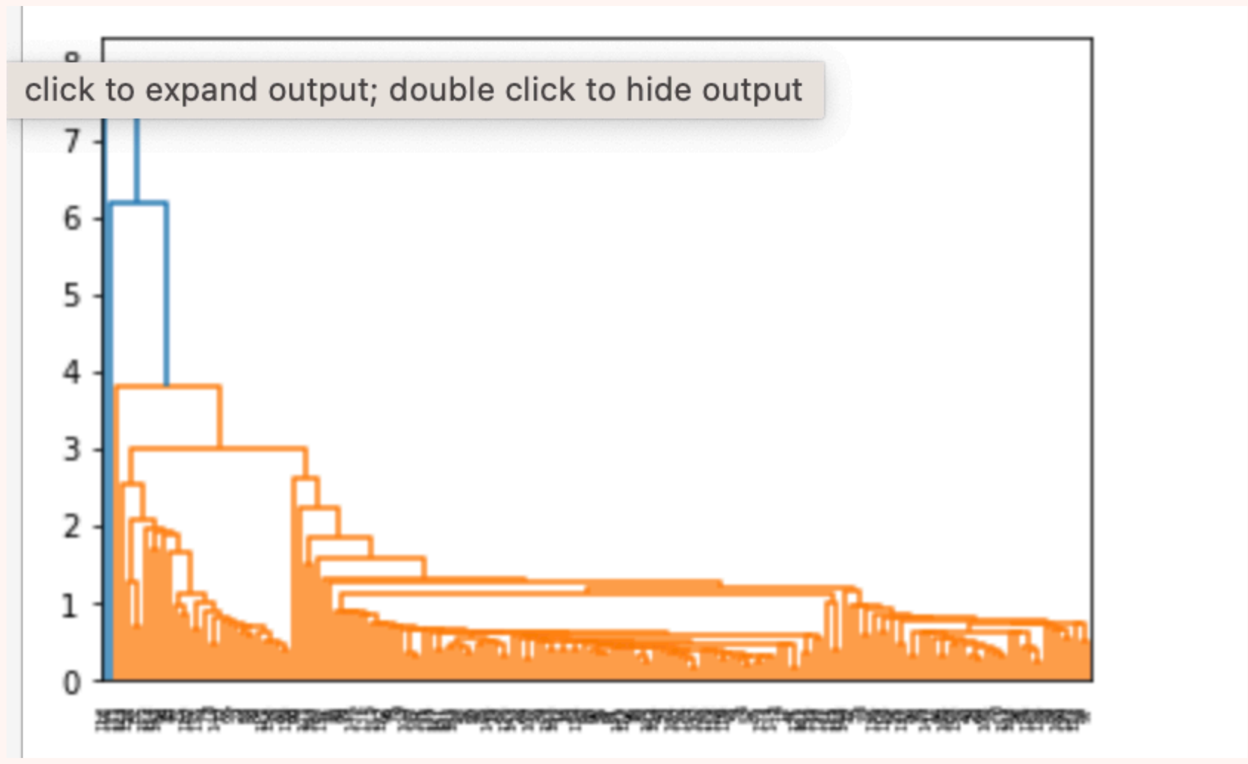
HIERARCHICAL CLUSTERING

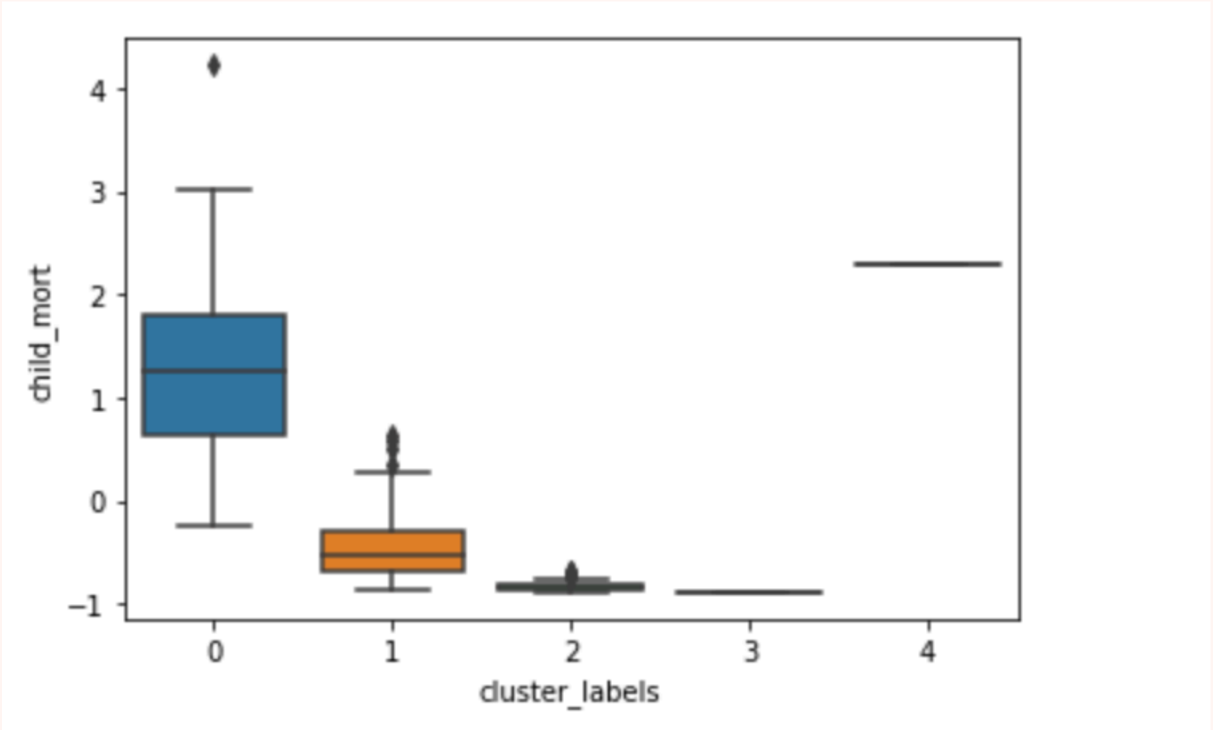
Hierarchical clustering was done using single and complete linkage. Also to observe this we can see label 0 has the highest child mortality, while lowest GDPP and income.











COUNTRIES CHOSEN FOR AID

We used the label 0 for hierarchical clustering and label 1 for k mean. Using sort function we found the highest mortality rate values and lowest GDPP and income.

country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	labels
Burundi	93.6	20.6052	26.7960	90.552	764.0	12.30	57.7	6.26	231.0	1
Liberia	89.3	62.4570	38.5860	302.802	700.0	5.47	60.8	5.02	327.0	1
Congo, Dem. Rep.	116.0	137.2740	26.4194	165.664	609.0	20.80	57.5	6.54	334.0	1
Niger	123.0	77.2560	17.9568	170.868	814.0	2.55	58.8	7.49	348.0	1
Sierra Leone	160.0	67.0320	52.2690	137.655	1220.0	17.20	55.0	5.20	399.0	1