Python Project

on

Clustering for Prospective Aid

Submitted by

Ranjana Chakraverty

Registration No.: 12014011

Program Name: B.Tech. (CSE - Data Science (ML and AI))

Under the guidance of

Ved Prakash Chaubey

School of Computer Science and Engineering

Lovely Professional University, Phagwara

(August-November, 2023)

DECLARATION

I hereby declare that I have completed my Python project from 20th August 2023 to 20th November 2023 under the guidance of Ved Prakash Chaubey. I have worked with full dedication during these twelve weeks and my learning outcomes fulfil the requirements of training for the award of degree of B.Tech. (CSE - Data Science (ML and AI)), Lovely Professional University, Phagwara.

Ranjana Chakraverty

Registration Number: 12014011

Date: 20th November 2023

ACKNOWLEDGEMENT

I would like to thank my professors at LPU of the School of Computer Science and Engineering for their support. I would like to thank the instructors at Upgrad, especially Ved Prakash Chaubey, who was very helpful and always available to give clear my doubts.

TABLE OF CONTENTS

OBJECTIVE AND SCOPE

Many international organisations like the UN, the IMF or the World Bank routinely provide aid to countries in the form of goods or just money to bail them out of crisis situations. Most countries that receive such aid are low-income countries with improper governance. But, providing aid raises a critical question. Which countries truly deserve the aid. The number of benefits provided to a country are huge, and are prone to be misused or mismanaged due to the rampant corruption and political instability in the regions they go to.

Obviously, most aid organisations must be using an elaborate system to determine which countries deserve aid, based on their previous track record of aid. While it can take hours to pour over long documents from yesteryears detailing the parameters of a certain country, machine learning, when brought in, has the capability to speed up this task enormously.

The endeavour of this assignment is to mimic an ML algorithm that such an aid organisation might use to categorise countries into different segments so that they may choose a certain group to fund.

The novelty of this problem is that it may seem like a classification problem but it is actually more sophisticated than that. The goal is not to classify countries into two boxes, one of which deserved aid and one which does not. Instead, the aim is to group companies into different categories, and find a group of countries that are in most need of aid, and can use it appropriately.

Various socio-economic metrics will be used for the categorisation which will henceforth be referred to as clustering. With this, the goal will be to choose the cluster deserving of aid, and then provide aid to the countries.

The model I will build in this project can be used by many aid organisations. Someday, perhaps the model might be able to be more sophisticated by including more social factors, and it can be extrapolated for other purposes as well. This projective has a great scope for use in the real world as it incorporated data from actual countries; it is a genuine dataset and not a synthesised one.

INTRODUCTION

Foreign aid refers to the international movement of money, services, or goods from governments or international institutions for the benefit of the receiving country or its citizens. Foreign aid can be fiscal, military, or humanitarian and is considered one of the significant sources of foreign exchange.

It is the voluntary movement of money or other resources from one nation to another. The transactions are mostly from developed countries to developing countries. A developing nation typically lacks a strong manufacturing base and is distinguished by a low value of the Human Development Index (HDI). Foreign aid may be offered as a contribution or a loan, which can either be a hard or soft loan. If the loan is in a foreign currency, it is termed as a hard loan.

- Foreign aid is the voluntary movement of resources from one country to another.
- Foreign aid may require the transfer of professional advice and training, commodities, or financial resources.
- The assistance can be used to advance the political aims of a government, allowing it to obtain diplomatic recognition.

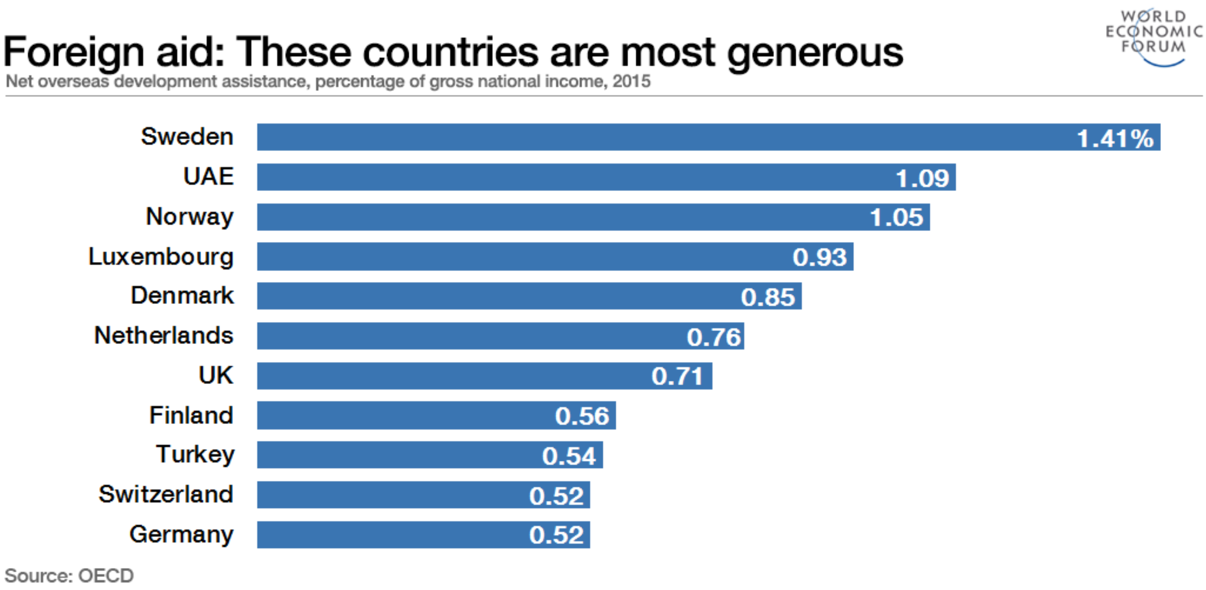
Purpose

Foreign aid may require the transfer of professional advice and training, or commodities or financial resources. Financial resources can occur in the form of concessional loans or grants, such as export credits. Official Development Assistance (ODA) is the most common form of foreign aid, which is the help provided to support development and to fight poverty. The main source of ODA is the bilateral grants from one country to another, while some of the funding is in the form of loans, and often it is channelled by non-governmental organizations and foreign organizations.

Countries also offer foreign aid in order to improve their own security. Economic aid may also be used to discourage friendly countries from coming under the control of unfriendly governments or paying for the right to set up or use military bases on foreign soil.

Foreign aid can be used to accomplish the political aims of a government, allowing it to obtain diplomatic recognition, to gain respect for its role in international institutions, or to improve the accessibility of its diplomats to foreign countries.

Foreign aid also seeks to promote the exports of a country and spread its literature, culture, or religion. Countries often provide aid to relieve the distress caused by man-made or natural disasters like drought, illness, and conflict. It helps to promote sustainable prosperity, create or reinforce political institutions, and address a range of worldwide concerns, including cancer, terrorism, and other violations, and environmental degradation.



<u>Types</u>

1. Tied Aid

Tied aid is a type of foreign aid that must be invested in a country that is providing support or in a group of chosen countries. A developed country can offer a bilateral loan or grant to a developing nation but will be required by the government to invest the money on goods and services produced in that country.

2. Bilateral Aid

Bilateral aid is given directly by one country's government to that of another country's government. It occurs when money flows from a country with a developed economy to a country with a developing economy. Bilateral aid is directed by strategic, political, and humanitarian interests. This is meant to further foster democracy, economic growth, peace, and sustainability of long-term programs.

3. Multilateral Aid

Multilateral aid is the support offered by several countries that share funds to foreign organizations such as the United Nations, the World Bank, and the International Monetary Fund (IMF). The funds are used to relieve hunger in developing nations. While the sector represents a minority of financial aid of the U.S., the donations provided by the country make up a large proportion of the donor funds obtained by the organizations.

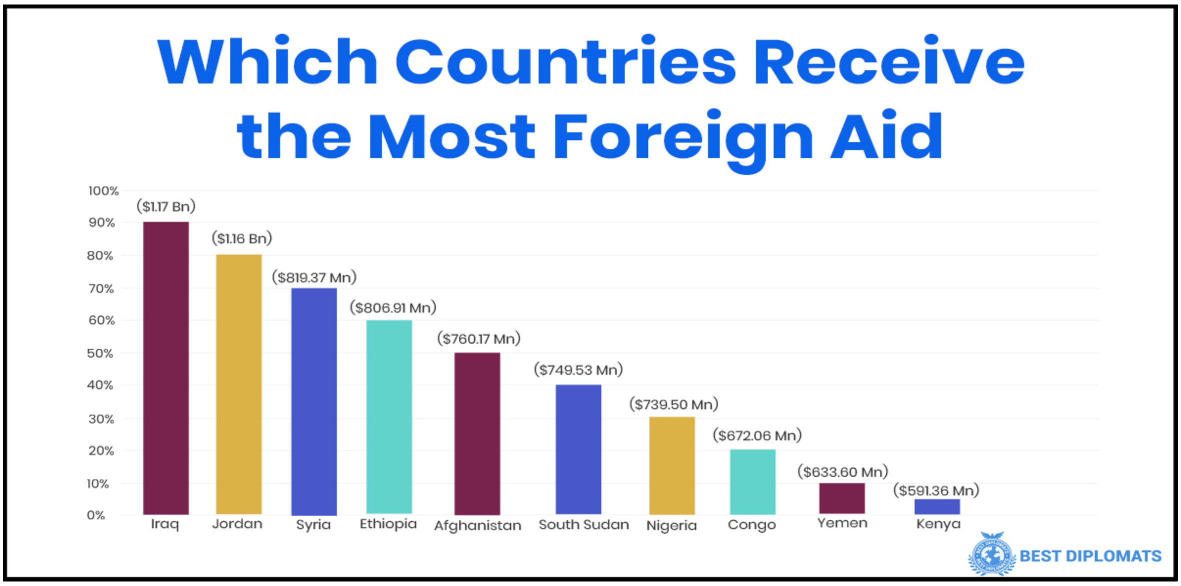
4. Military Aid

Military aid typically allows the recipient country either to procure weapons or security contracts directly from the U.S. In other situations, it actually simplifies the mechanism by enabling the federal government to buy weapons on its own and ship them to military transport.

5. Project Aid

The assistance is known to be project aid when the funds are used to support a certain project, such as a hospital or school.

Foreign aid has been used, particularly in poorer countries, to fund or to monitor elections, to facilitate judicial reforms, and to assist the activities of human rights organizations and labour groups. Foreign aid is also used to address transnational problems such as the production and export of illegal drugs and the battle against HIV/AIDS. Foreign assistance is still used to promote economic development. Countries often provide foreign aid to enhance their own security. Thus, economic assistance may be used to prevent friendly governments from falling under the influence of unfriendly ones or as payment for the right to establish or use military bases on foreign soil. Foreign aid also may be used to achieve a country's diplomatic goals, enabling it to gain diplomatic recognition, to garner support for its positions in international organizations, or to increase its diplomats' access to foreign officials. Other purposes of foreign aid include promoting a country's exports (e.g., through programs that require the recipient country to use the aid to purchase the donor country's agricultural products or manufactured goods) and spreading its language, culture, or religion. Countries also provide aid to relieve suffering caused by natural or man-made disasters such as famine, disease, and war, to promote economic development, to help establish or strengthen political institutions, and to address a variety of transnational problems including disease, terrorism and other crimes, and destruction of the environment.



Foreign aid is any type of assistance that one country's government provides to another nation, usually from developed to developing nations. Governments may issue aid in the form of:

- Money
- Food and supplies
- Medical assistance including doctors and supplies
- Humanitarian aid such as relief workers
- Training services including agricultural training
- Health care
- Education
- Assistance with infrastructure building
- Activities related to peacebuilding

PROFILE & ANALYSIS OF THE PROBLEM

The goal will be to categorise the countries given in a data set using some socio-economic and health factors that determine the overall development of the country.

The dataset used for this project has been sourced from Kaggle, and the link is below.

https://www.kaggle.com/datasets/subhajitnayak/country-data

It contains real data from 167 countries with various socio-economic factors in columns. These can be used to cluster the countries.

The columns in the dataset are:

- country (object)
- child_mort (numerical)
- exports (numerical)
- health (numerical)
- imports (numerical)
- income (numerical)
- inflation (numerical)
- life_expec (numerical)
- total_fer (numerical)
- gdpp (numerical)

Based on the above columns, the task is to build a clustering-based machine learning model that can categorise countries into different clusters based on these columns. The countries that have the worst record in the metrics will be provided with aid.

The results can be used by any international humanitarian NGO committed to fighting poverty and providing the people of backward countries with basic amenities and relief during the time of disasters and natural calamities. They can regularly run operational projects, along with advocacy drives, to raise awareness and for funding purposes.

METHOD

The following broad approach will be followed for this assignment.

- Data Inspection
- EDA
- Outlier analysis
- K-Means Clustering
- Hierarchical Clustering
- Cluster Analysis
- Visualisations
- Listing results in the chosen cluster

The columns can be described as follows:

- Country: name of the country
- Child_mort: death of children under 5 years of age per 1000 live births
- Exports: exports of goods and serviced per capita, given as percentage of the GDP
- Health: total health spending per capita, given as percentage of the GDP
- Imports: imports of goods and serviced per capita, given as percentage of the GDP
- Income: net income per person
- Inflation: the measurement of the annual growth rate of the total GDP
- Life_expec: the average number of years a new born child would live if the current mortality patterns are to remain the same
- Total_fer: the number of children that would be born to each woman if the current age fertility rate remains the same
- Gdpp: the GDP per capita, calculated as the total GDP divided by the total population

The python notebook can be found at the end of this document.

RESULT

K-Means

For deciding the number of clusters, the silhouette score was used.

5 was chosen as k's value but it yielded 2 redundant clusters. So, the k value was reduced to 3. With this, proper clusters were obtained. The performance of each was evaluated with respect to gdpp, income and child mortality with the help of boxplots. Cluster 1 had the worst performance overall. It has a total of 48 countries. Out of these 48 countries, the top 10 worst performers were chosen for prospective aid.

Hierarchical

Single linkage was first used and it was unable to yield proper clusters. To combat this, complete linkage was used which output 3 clusters. With the help of visualisations, it was determined that Cluster 0, with 148 countries had the worst performance, which is obviously too many. So, the top 10 worst performers were chosen out of this cluster as well to provided aid to.

What is worth noticing here is that K-Means and Hierarchical clustering differed significantly in the clusters they created, but the top 10 values chosen out of each clustering technique were the same, further indicating that these are the only countries that deserve aid seeing as they were common in both the clustering techniques.

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

BIBLIOGRAPHY

https://www.weforum.org/

https://www.oecd.org

https://bestdiplomats.org

corporatefinanceinstitute.com

britannica.com

investopedia.com

In [1]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

from sklearn.preprocessing import StandardScaler

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
```

In [2]:

```
df = pd.read_csv("Country-data.csv")
```

Data Inspection & EDA

In [3]:

```
df.head()
```

Out[3]:

	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 [4]:

```
df.shape
```

Out[4]:

```
(167, 10)
```

In [5]:

```
df.info()
```

```
<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
```

In [6]:
```
df.describe()
```

Out[6]:

	child_mort	exports	health	imports	income	inflation	life_expec	tota
count	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.00
mean	38.270060	41.108976	6.815689	46.890215	17144.688623	7.781832	70.555689	2.94
std	40.328931	27.412010	2.746837	24.209589	19278.067698	10.570704	8.893172	1.51
min	2.600000	0.109000	1.810000	0.065900	609.000000	-4.210000	32.100000	1.15
25%	8.250000	23.800000	4.920000	30.200000	3355.000000	1.810000	65.300000	1.79
50%	19.300000	35.000000	6.320000	43.300000	9960.000000	5.390000	73.100000	2.41
75%	62.100000	51.350000	8.600000	58.750000	22800.000000	10.750000	76.800000	3.88
max	208.000000	200.000000	17.900000	174.000000	125000.000000	104.000000	82.800000	7.49

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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
```

As per the data description, three columns are given as percentage of the GDP, and here I'll convert them back into decimal (base 10) values.

In [8]:
```
df['exports'] = (df['exports'] * df['gdpp']) / 100
df['health'] = (df['health'] * df['gdpp']) / 100
df['imports'] = (df['imports'] * df['gdpp']) / 100
```

In [9]:
```
df.head()
```

Out[9]:

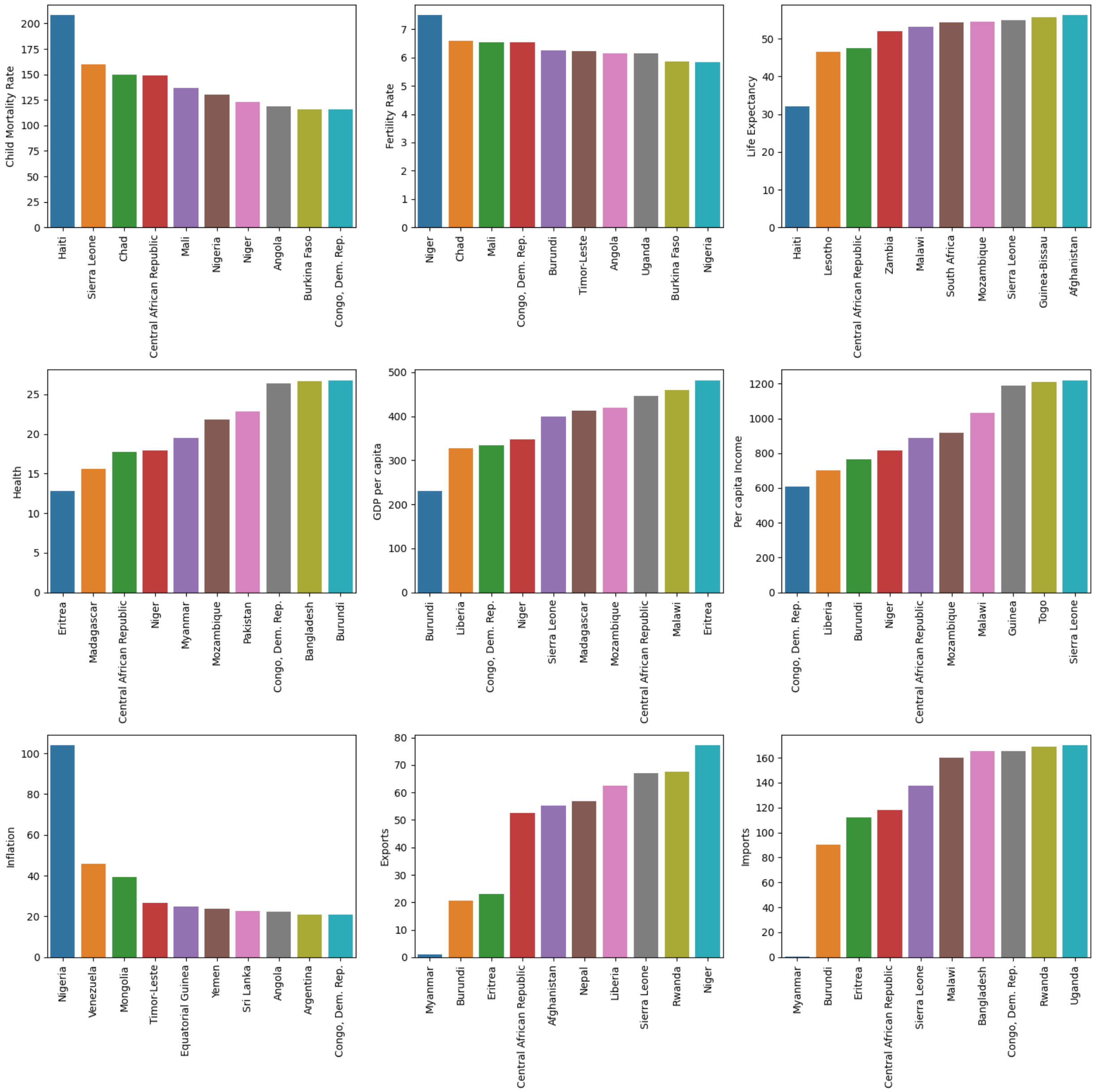
	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdp
0	Afghanistan	90.2	55.30	41.9174	248.297	1610	9.44	56.2	5.82	55
1	Albania	16.6	1145.20	267.8950	1987.740	9930	4.49	76.3	1.65	409
2	Algeria	27.3	1712.64	185.9820	1400.440	12900	16.10	76.5	2.89	446
3	Angola	119.0	2199.19	100.6050	1514.370	5900	22.40	60.1	6.16	353
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100	1.44	76.8	2.13	1220

As we can see, the columns are now back to normal values, instead of percentages.

Plotting those countries which have the 10 lowest values in all the columns.

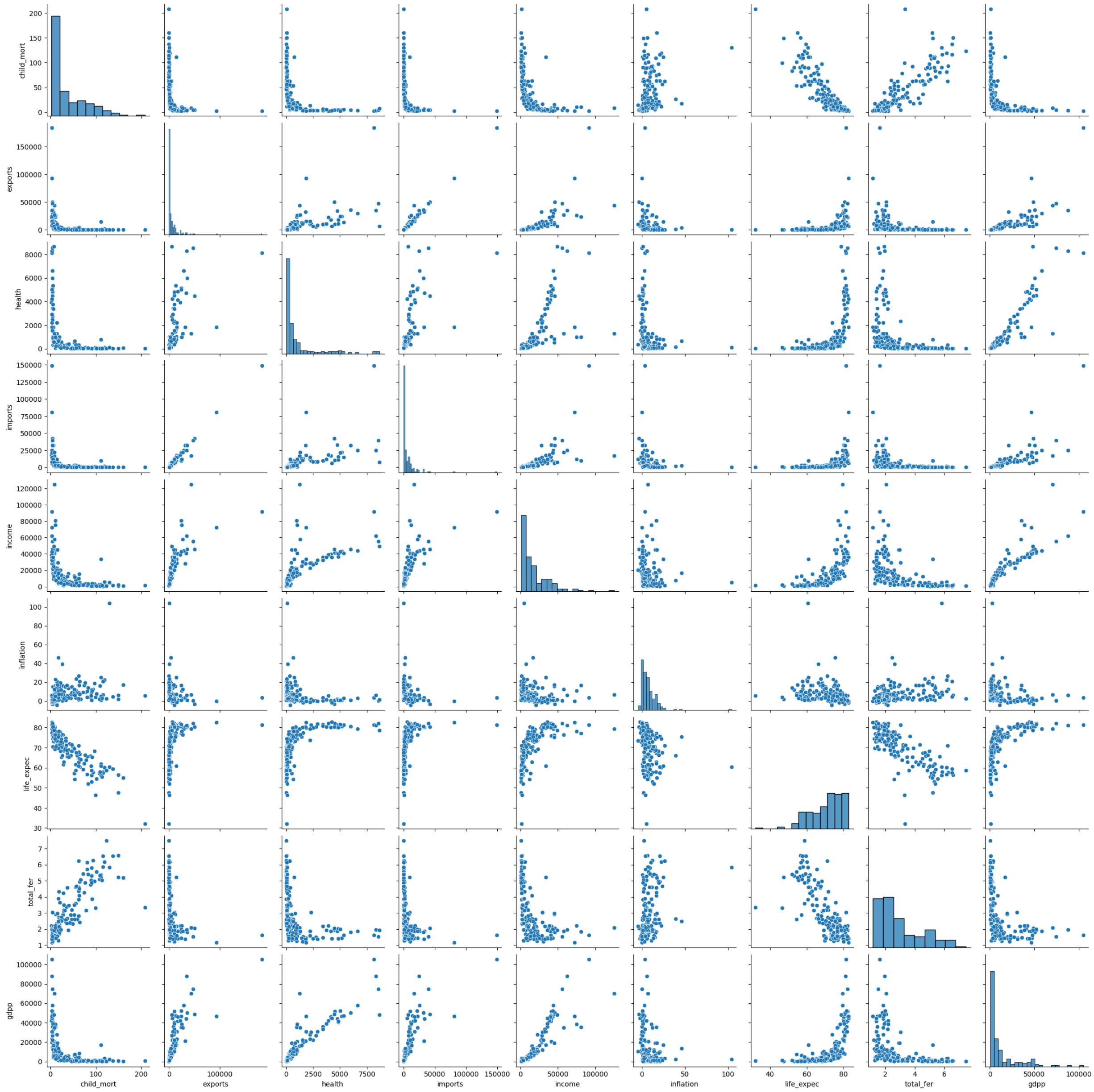
In [10]:

```
fig, axs = plt.subplots(3,3,figsize = (15,15))



top10_child_mort = df[['country','child_mort']].sort_values('child_mort', ascending
plt1 = sns.barplot(x='country', y='child_mort', data= top10_child_mort, ax = axs[0,0
plt1.set(xlabel = '', ylabel= 'Child Mortality Rate')

top10_total_fer = df[['country','total_fer']].sort_values('total_fer', ascending = F
plt1 = sns.barplot(x='country', y='total_fer', data= top10_total_fer, ax = axs[0,1])
plt1.set(xlabel = '', ylabel= 'Fertility Rate')



bottom10_life_expec = df[['country','life_expec']].sort_values('life_expec', ascendi
plt1 = sns.barplot(x='country', y='life_expec', data= bottom10_life_expec, ax = axs[
plt1.set(xlabel = '', ylabel= 'Life Expectancy')


bottom10_health = df[['country','health']].sort_values('health', ascending = True).h
plt1 = sns.barplot(x='country', y='health', data= bottom10_health, ax = axs[1,0])
plt1.set(xlabel = '', ylabel= 'Health')
bottom10_gdpp = df[['country','gdpp']].sort_values('gdpp', ascending = True).head(10
plt1 = sns.barplot(x='country', y='gdpp', data= bottom10_gdpp, ax = axs[1,1])
plt1.set(xlabel = '', ylabel= 'GDP per capita')

bottom10_income = df[['country','income']].sort_values('income', ascending = True).h
plt1 = sns.barplot(x='country', y='income', data= bottom10_income, ax = axs[1,2])
plt1.set(xlabel = '', ylabel= 'Per capita Income')



top10_inflation = df[['country','inflation']].sort_values('inflation', ascending = F
plt1 = sns.barplot(x='country', y='inflation', data= top10_inflation, ax = axs[2,0])
plt1.set(xlabel = '', ylabel= 'Inflation')
bottom10_exports = df[['country','exports']].sort_values('exports', ascending = True
plt1 = sns.barplot(x='country', y='exports', data= bottom10_exports, ax = axs[2,1])
plt1.set(xlabel = '', ylabel= 'Exports')


bottom10_imports = df[['country','imports']].sort_values('imports', ascending = True
plt1 = sns.barplot(x='country', y='imports', data= bottom10_imports, ax = axs[2,2])
plt1.set(xlabel = '', ylabel= 'Imports')

for ax in fig.axes:
    plt.sca(ax)
    plt.xticks(rotation = 90)

plt.tight_layout()
plt.show()
```



In [11]:

```
sns.pairplot(df)
plt.show()
```

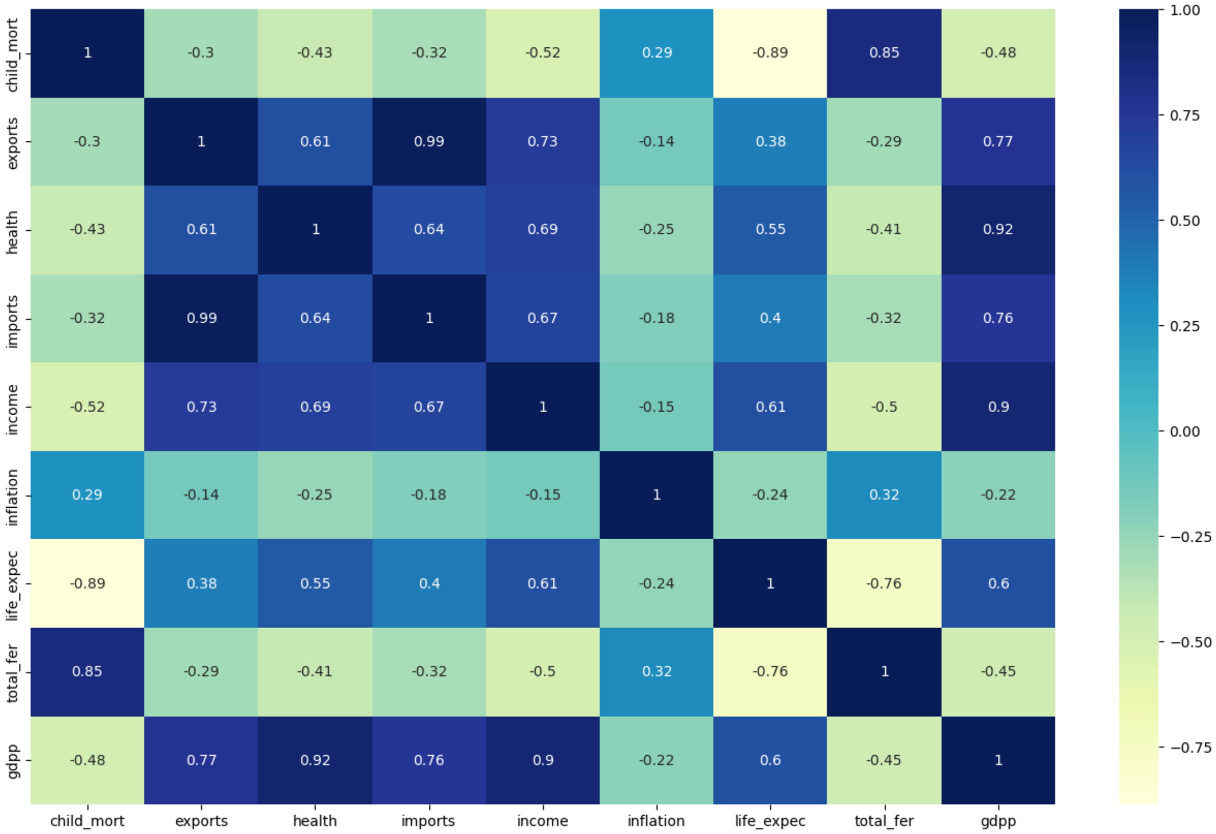


Plotting the correlation matrix to check for multicollinearity.

In [12]:

```
plt.figure(figsize = (16, 10))
sns.heatmap(df.corr(), annot = True, cmap="YlGnBu")
plt.show()
```

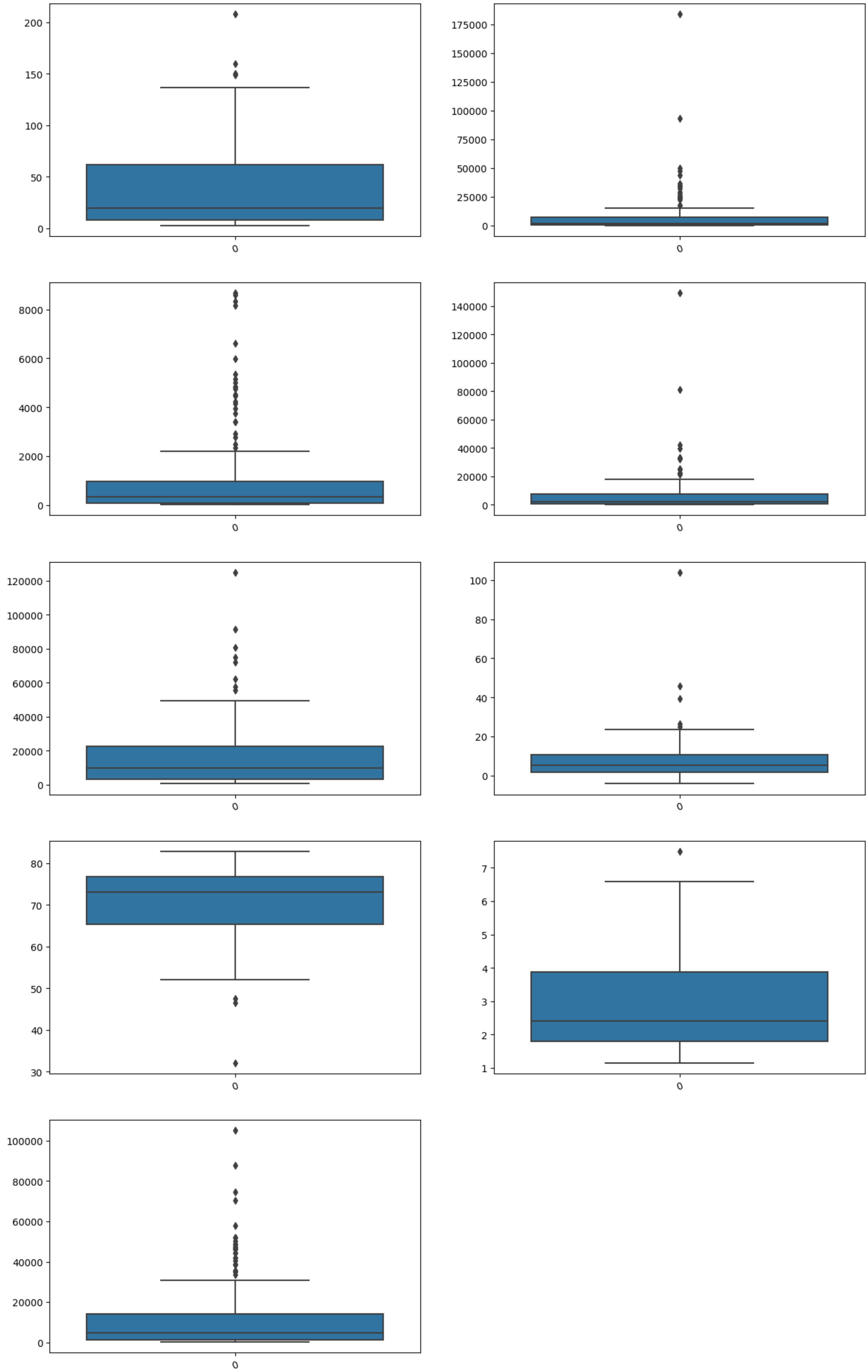
```
<ipython-input-12-dbfad3d97249>:2: FutureWarning: The default value of numeric_only i
n DataFrame.corr is deprecated. In a future version, it will default to False. Select
only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), annot = True, cmap="YlGnBu")
```



Outlier Analysis

In [13]:

```
colo= ['child_mort','exports','health','imports','income','inflation','life_expec','
plt.figure(figsize=(15,25))
for i in enumerate(colo):
    ax = plt.subplot(5, 2, i[0]+1)
    sns.boxplot(df[i[1]])
    plt.xticks(rotation = 20)
plt.show()
```



In the preceding cell, I created a box plot for all the columns. There are multiple countries whose GDP is extremely high. These are probably the developed countries where the quality of life is excellent. It is pertinent to realise that in this case of clustering countries according to their economic needs, removing outliers is not advisable. If the countries with extremely high values of child mortality are removed, then they will be ineligible for help from humanitarian organisations which makes this whole assignment redundant. So, considering the unique circumstance, in this dataset, outliers will be allowed to remain.

Clustering Model

Preprocessing

In [14]:
```
# hopkin's 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).r
        ujd.append(u_dist[0][1])
        w_dist, _ = nbrs.kneighbors(X.iloc[rand_X[j]].values.reshape(1, -1), 2, retu
        wjd.append(w_dist[0][1])

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

    return H
```

In [15]:
```
hopkins(df.drop('country',axis=1))
```

Out[15]:
```
0.9532941352093782
```

Thankfully, the data is highly clusterable.

In [16]:
```
#scaling
dfx = df.drop('country', axis = 1)
scale = StandardScaler()
dfx = scale.fit_transform(dfx)
```

K-Means

In [17]:
```
range_n_clusters = [2, 3, 4, 5, 6, 7, 8]

for num_clusters in range_n_clusters:
```

```
    # intialise kmeans
    kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(dfx)

    cluster_labels = kmeans.labels_

    # silhouette score
    silhouette_avg = silhouette_score(dfx, cluster_labels)
    print("For n_clusters={0}, the silhouette score is {1}".format(num_clusters, sil
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
For n_clusters=2, the silhouette score is 0.45863306035476264
For n_clusters=3, the silhouette score is 0.4218615812599681
For n_clusters=4, the silhouette score is 0.42914711278370843
For n_clusters=5, the silhouette score is 0.4324001169216119
For n_clusters=6, the silhouette score is 0.2908984109903817
For n_clusters=7, the silhouette score is 0.3065505636750877
For n_clusters=8, the silhouette score is 0.3075751716874681
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
```

So, the number of clusters will be 5.

In [18]:

```
kmeans = KMeans(n_clusters=5, max_iter=100 , random_state = 100)
kmeans.fit(dfx)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
```

Out[18]:

```
▾                     KMeans
KMeans(max_iter=100, n_clusters=5, random_state=100)
```

In [19]:

```
kmeans.labels_
```

Out[19]:

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

```
        0, 1, 2, 1, 1, 1, 0, 2, 2, 2, 1, 2, 1, 1, 0, 0, 2, 1, 0, 1, 1, 0,
        0, 1, 1, 4, 1, 0, 0, 1, 1, 0, 2, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
        2, 2, 0, 3, 2, 1, 0, 1, 1, 1, 1, 1, 1, 2, 1, 1, 0, 1, 1, 0, 1, 1,
        0, 2, 1, 2, 0, 0, 1, 2, 1, 1, 0, 1, 2, 2, 1, 0, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 2, 2, 2, 1, 1, 1, 1, 1, 0, 0], dtype=int32)
```

In [20]:

```
df_km = pd.concat([df, pd.Series(kmeans.labels_)], axis = 1)
df_km.columns = ['country','child_mort','exports','health','imports','income','infla
df_km.head()
```

Out[20]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdp
0	Afghanistan	90.2	55.30	41.9174	248.297	1610	9.44	56.2	5.82	55
1	Albania	16.6	1145.20	267.8950	1987.740	9930	4.49	76.3	1.65	409
2	Algeria	27.3	1712.64	185.9820	1400.440	12900	16.10	76.5	2.89	446
3	Angola	119.0	2199.19	100.6050	1514.370	5900	22.40	60.1	6.16	353
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100	1.44	76.8	2.13	1220

In [21]:

```
df_km['cluster_id'].value_counts()
```

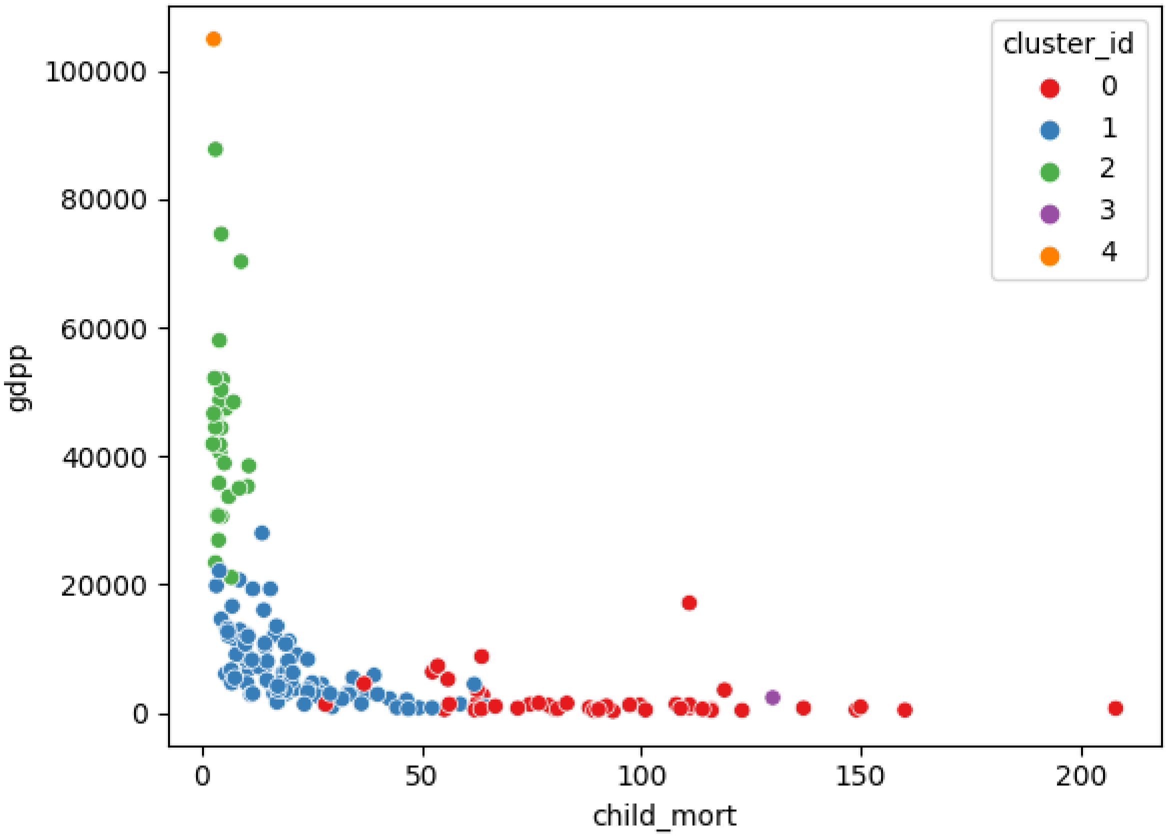
Out[21]:

```
1    88
0    47
2    30
4     1
3     1
Name: cluster_id, dtype: int64
```

Cluster Visualisation

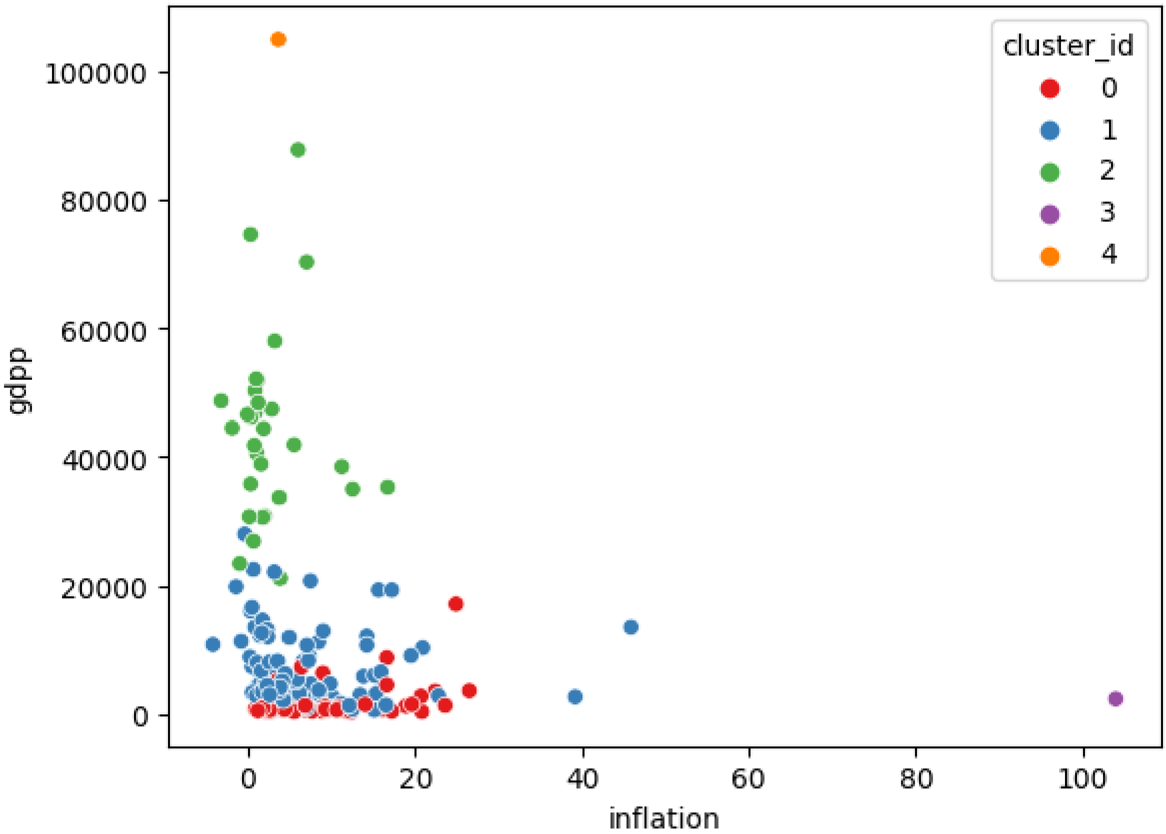
In [22]:

```
sns.scatterplot(x = 'child_mort', y = 'gdpp', hue ='cluster_id', legend = 'full', da
plt.show()
```



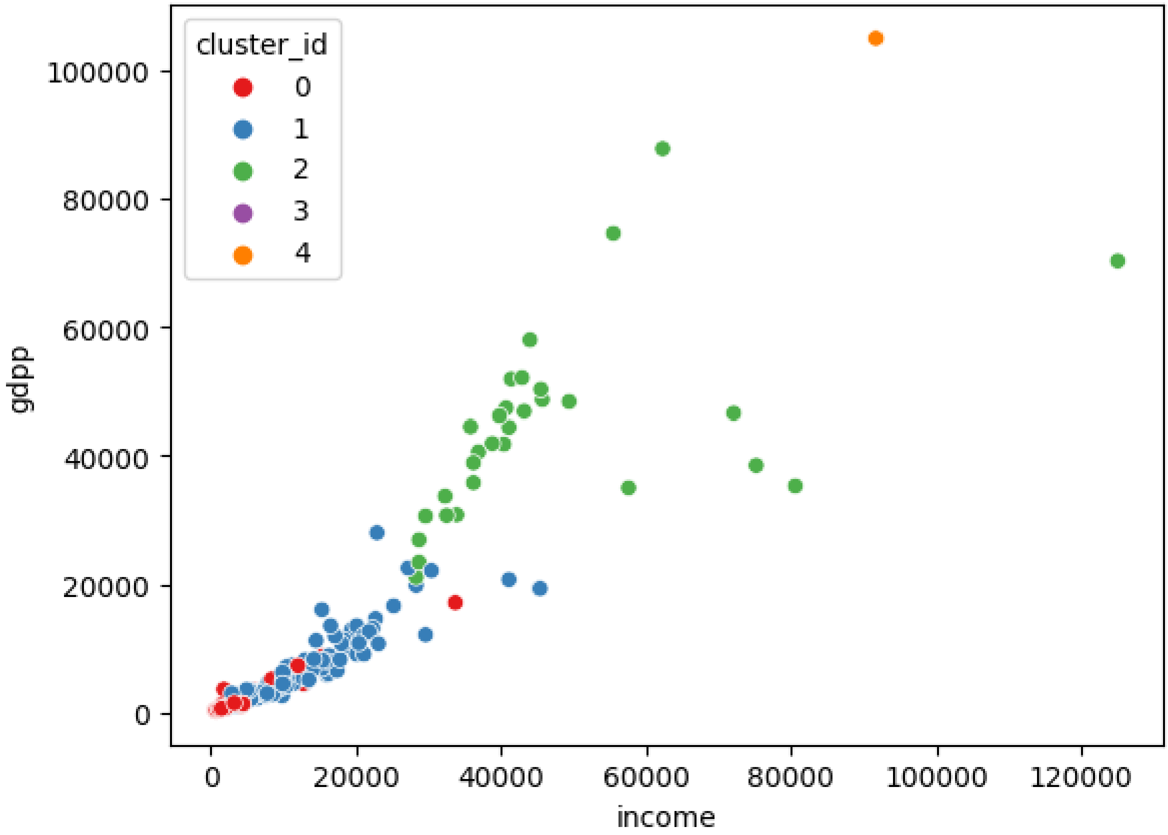
In [23]:

```
sns.scatterplot(x = 'inflation', y = 'gdpp', hue ='cluster_id', legend = 'full', dat
plt.show()
```

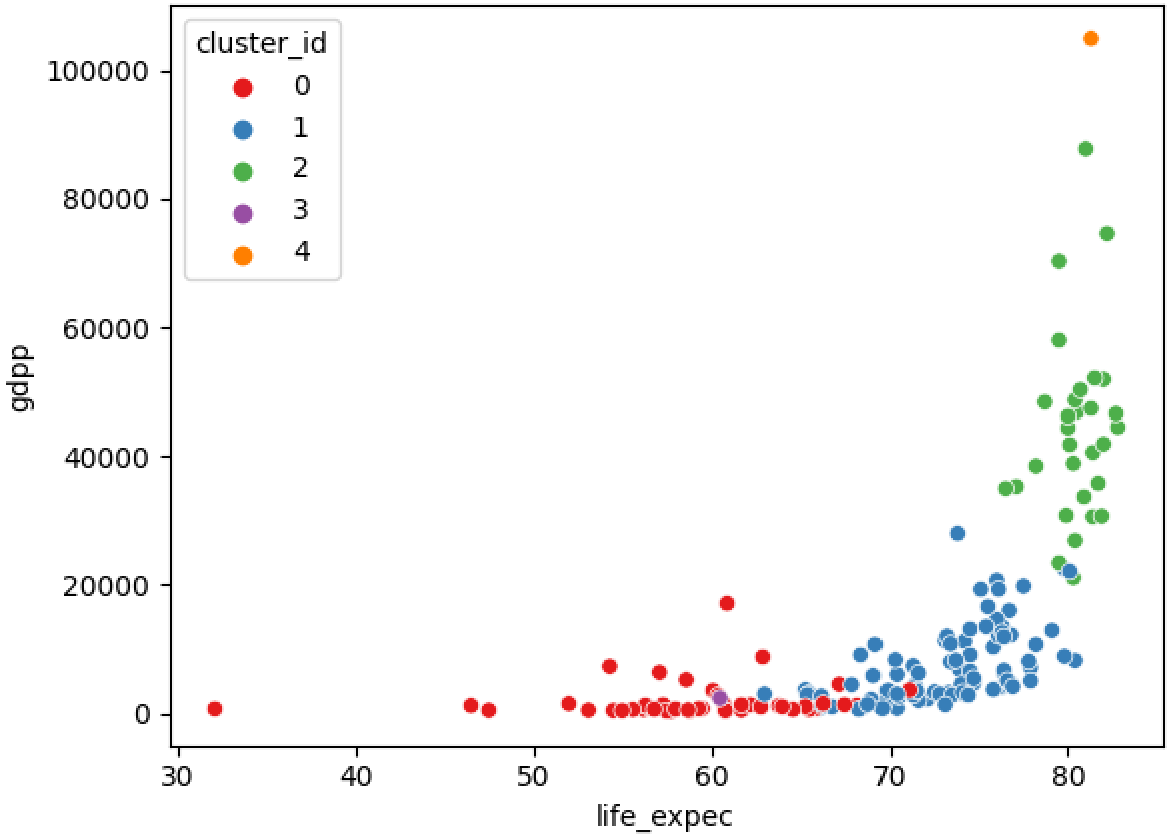


In [24]:

```
sns.scatterplot(x = 'income', y = 'gdpp', hue ='cluster_id', legend = 'full', data =
plt.show()
```



In [25]:
```
sns.scatterplot(x = 'life_expec', y = 'gdpp', hue ='cluster_id', legend = 'full', da
plt.show()
```



The orange and purple clusters are redundant, clearly. So, I'll set k equal to 3.

In [26]:
```
kmeans = KMeans(n_clusters = 3, max_iter=100 , random_state = 100)
```

```
kmeans.fit(dfx)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  warnings.warn(
```

Out[26]:

```
KMeans

KMeans(max_iter=100, n_clusters=3, random_state=100)
```

In [27]:

```
kmeans.labels_
```

Out[27]:

```
array([1, 2, 2, 1, 2, 2, 2, 0, 0, 2, 2, 2, 2, 2, 2, 0, 2, 1, 2, 2, 2, 1,
       2, 0, 2, 1, 1, 2, 1, 0, 2, 1, 1, 2, 2, 2, 1, 1, 1, 2, 1, 2, 0, 2,
       0, 2, 2, 2, 2, 1, 1, 2, 2, 0, 0, 1, 1, 2, 0, 1, 2, 2, 2, 1, 1, 2,
       1, 2, 0, 2, 2, 2, 1, 0, 2, 0, 2, 0, 2, 2, 1, 1, 0, 2, 1, 2, 2, 1,
       1, 2, 2, 0, 2, 1, 1, 2, 2, 1, 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2,
       0, 0, 1, 1, 0, 2, 1, 2, 2, 2, 2, 2, 2, 0, 2, 2, 1, 2, 2, 1, 2, 2,
       1, 0, 2, 2, 1, 1, 2, 0, 2, 2, 1, 2, 0, 0, 2, 1, 2, 1, 1, 2, 2, 2,
       2, 1, 2, 0, 0, 0, 2, 2, 2, 2, 2, 1, 1], dtype=int32)
```

In [28]:

```
df_km1 = pd.concat([df, pd.Series(kmeans.labels_)], axis = 1)
df_km1.columns = ['country','child_mort','exports','health','imports','income','infl
df_km1.head()
```

Out[28]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdp
0	Afghanistan	90.2	55.30	41.9174	248.297	1610	9.44	56.2	5.82	55
1	Albania	16.6	1145.20	267.8950	1987.740	9930	4.49	76.3	1.65	409
2	Algeria	27.3	1712.64	185.9820	1400.440	12900	16.10	76.5	2.89	446
3	Angola	119.0	2199.19	100.6050	1514.370	5900	22.40	60.1	6.16	353
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100	1.44	76.8	2.13	1220

In [29]:

```
df_km1['cluster_id'].value_counts()
```

Out[29]:

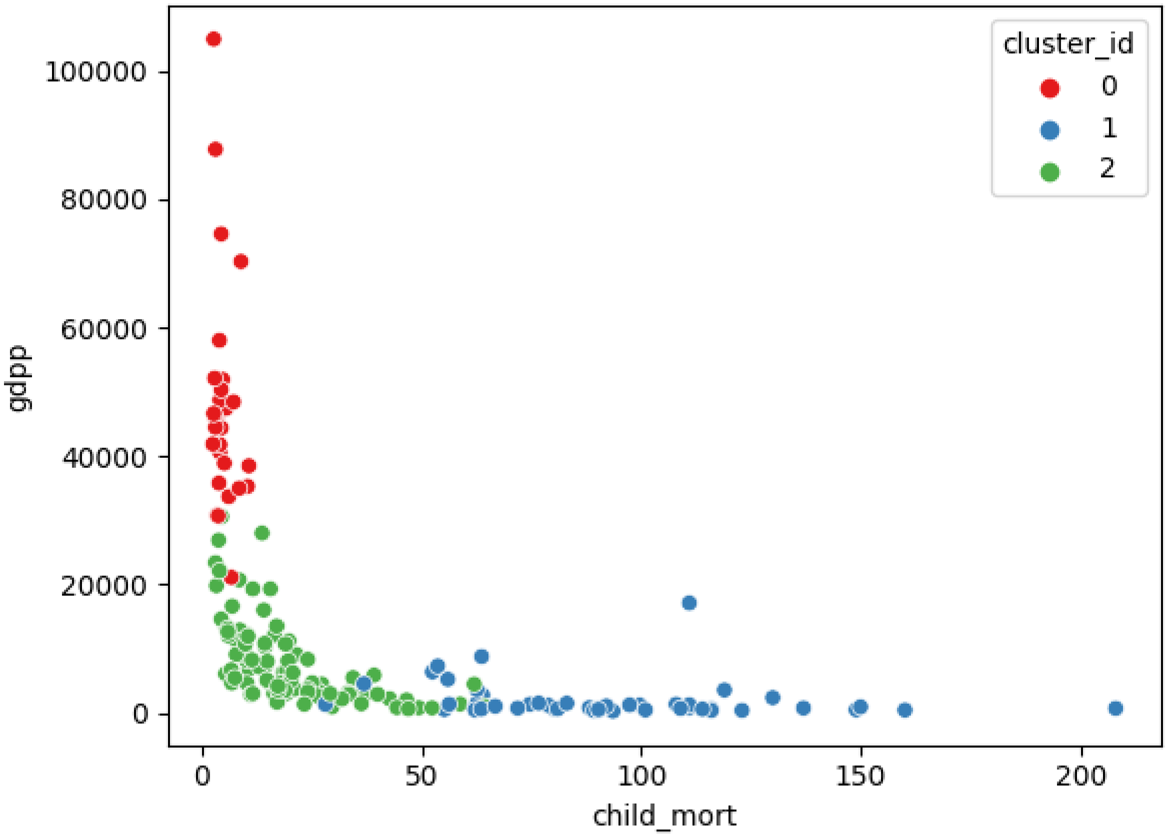
```
2    91
1    48
0    28
Name: cluster_id, dtype: int64
```

This looks much better than the previus k value.

Cluster Visualisation

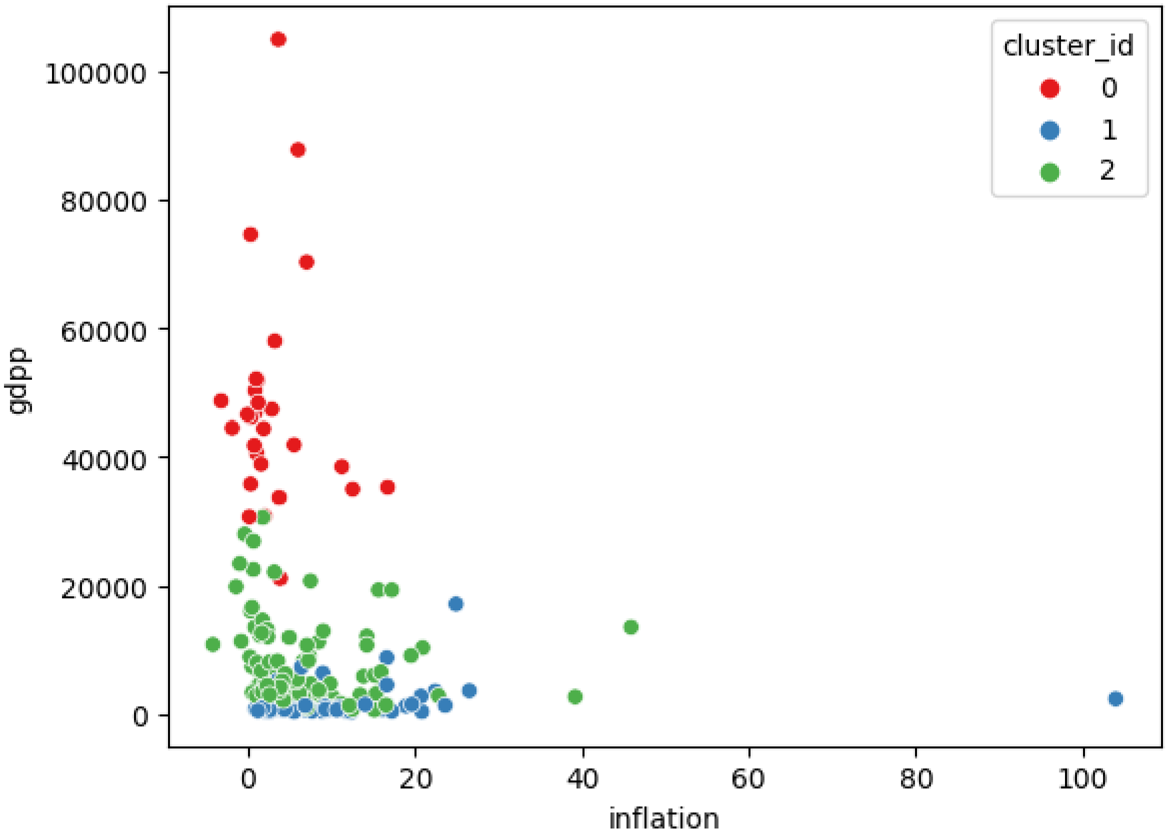
In [30]:

```
sns.scatterplot(x = 'child_mort', y = 'gdpp', hue ='cluster_id', legend = 'full', da
plt.show()
```



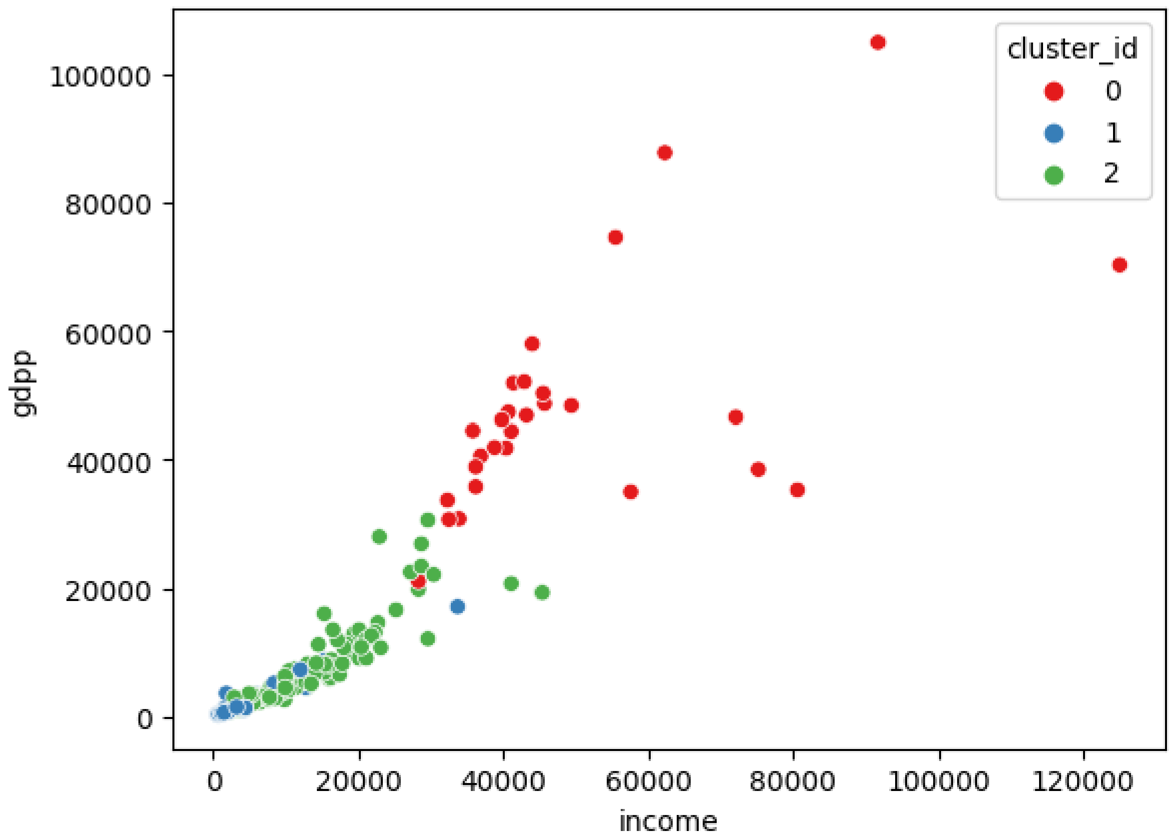
In [31]:

```
sns.scatterplot(x = 'inflation', y = 'gdpp', hue ='cluster_id', legend = 'full', dat
plt.show()
```



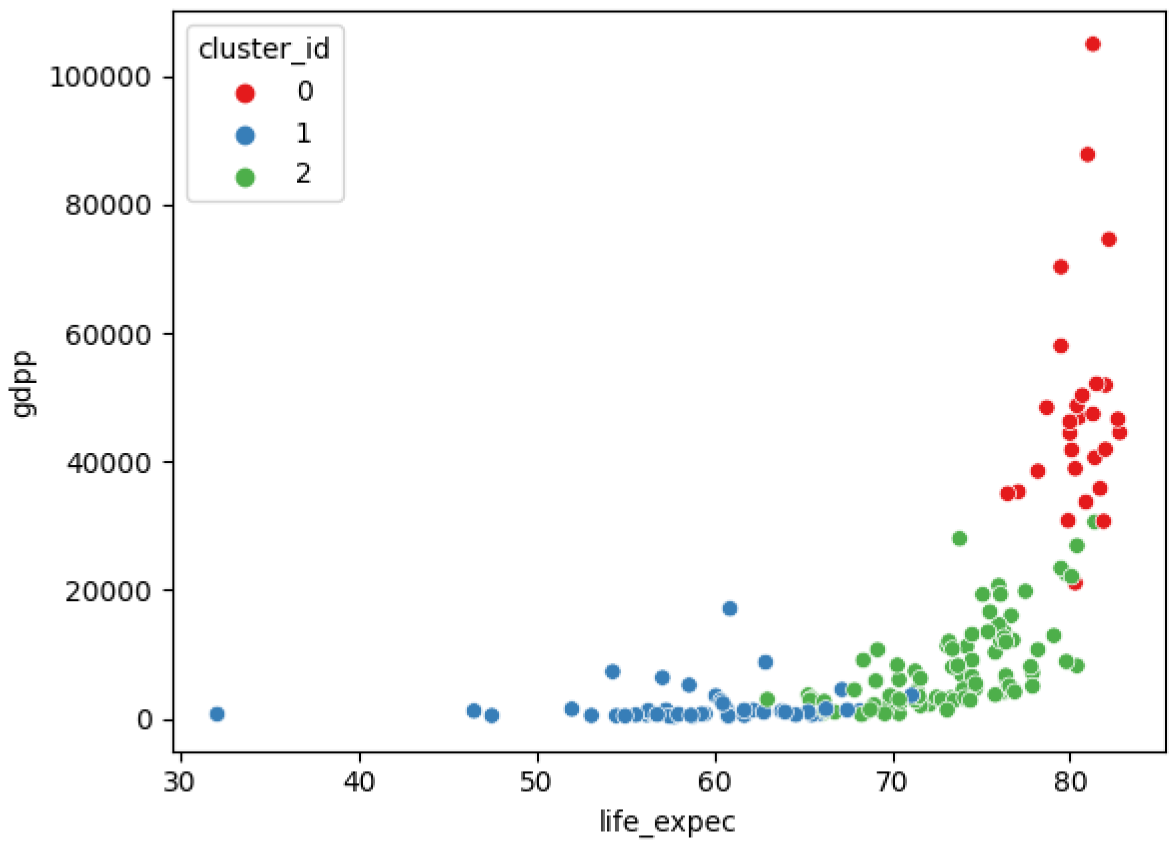
In [32]:

```
sns.scatterplot(x = 'income', y = 'gdpp', hue ='cluster_id', legend = 'full', data =
plt.show()
```



In [33]:

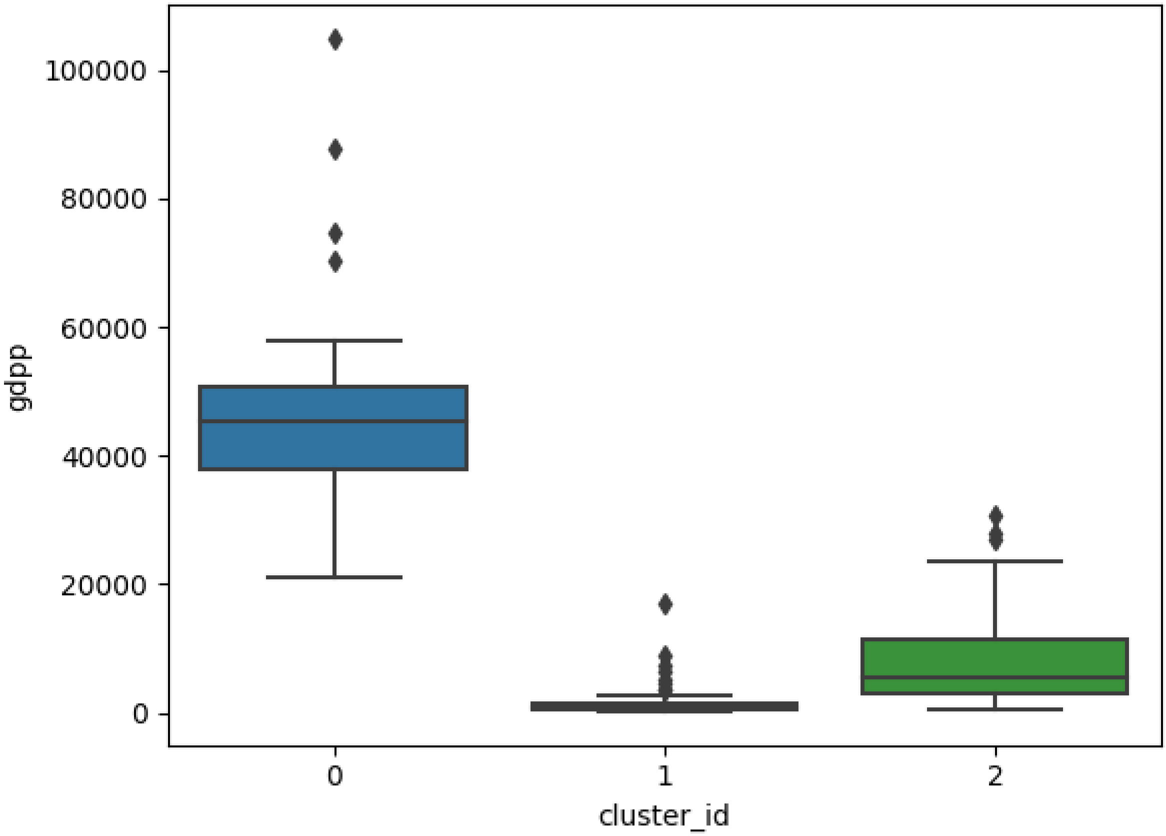
```
sns.scatterplot(x = 'life_expec', y = 'gdpp', hue ='cluster_id', legend = 'full', da
plt.show()
```



Here, we'll be using boxplots to confirm that the metrics for a particular cluster ID are particularly poor. Then this cluster can be chosen for aid.

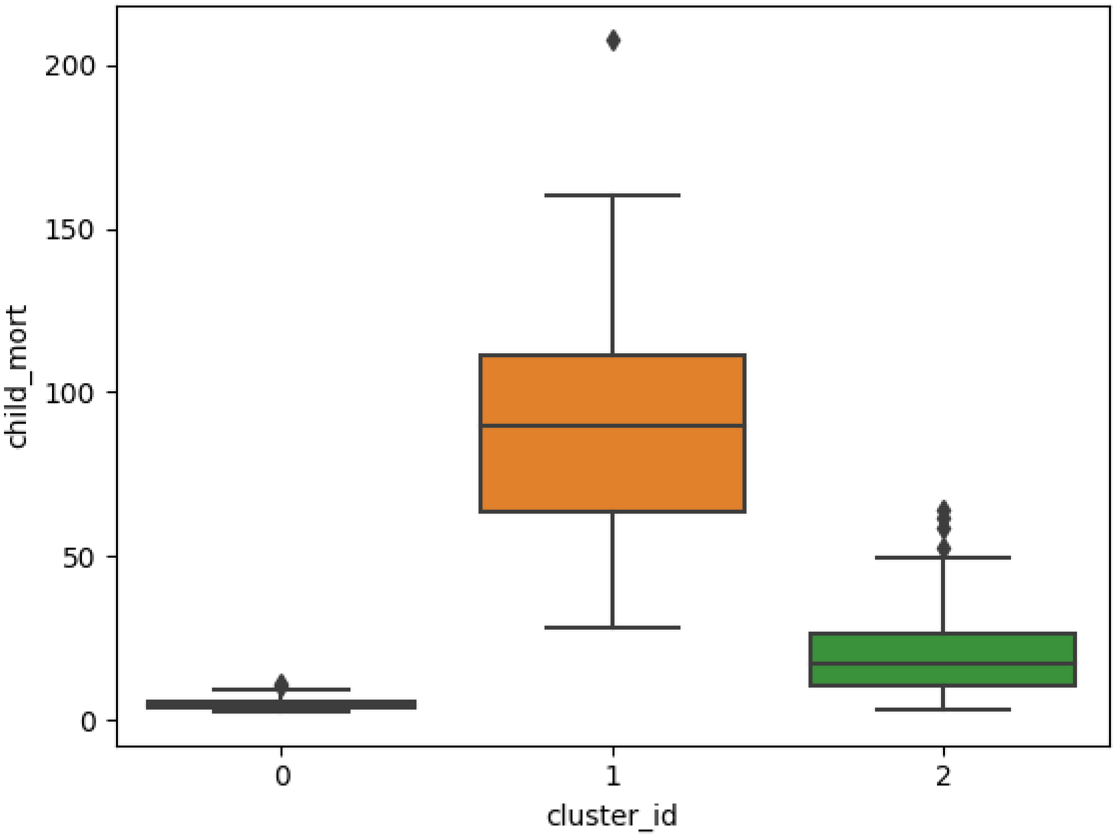
In [34]:

```
sns.boxplot(data = df_km1, x = 'cluster_id', y = 'gdpp')
plt.show()
```

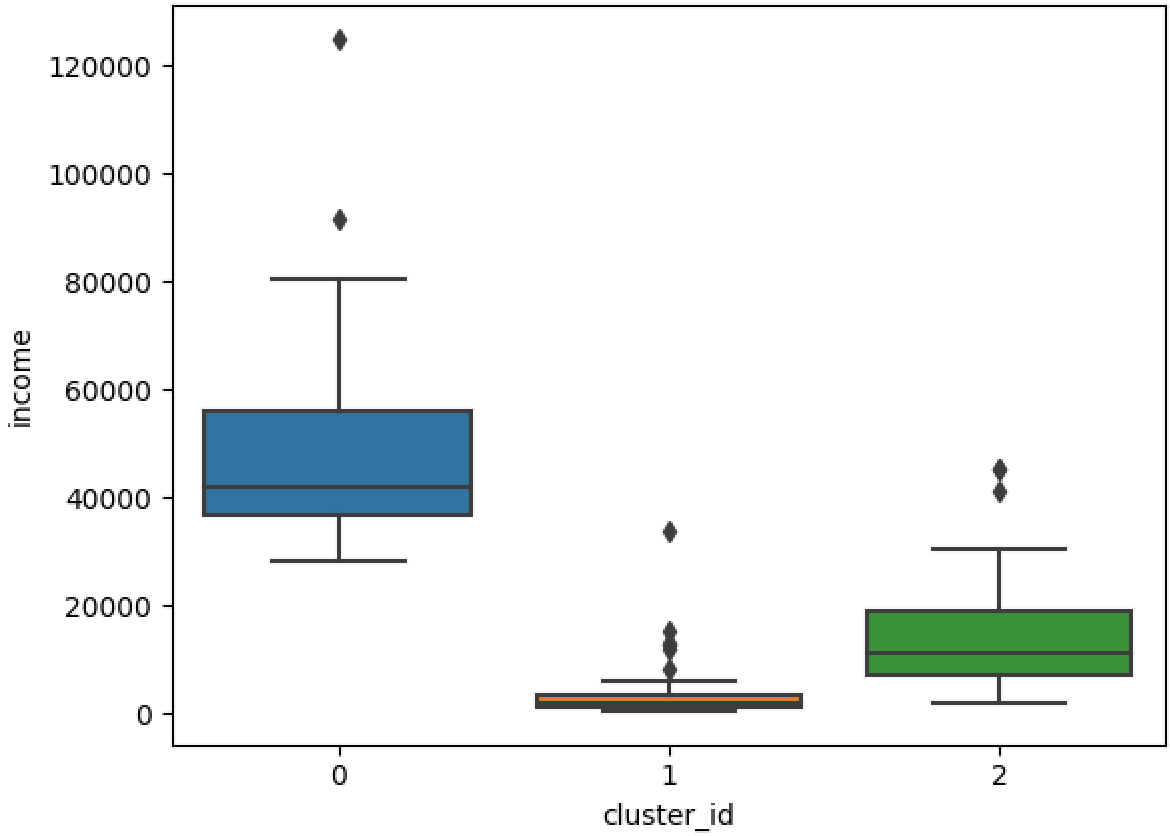


In [35]:

```
sns.boxplot(data = df_km1,x = 'cluster_id',y = 'child_mort')
plt.show()
```



In [36]:
```
sns.boxplot(data = df_km1, x = 'cluster_id',y = 'income')
plt.show()
```



Clearly, cluster 1 has the worst performance. It has low income and GDP but high child mortality. It is therefore the worthy candidate for aid.

In [37]:
```
df_km1[df_km1['cluster_id'] == 1]['country']
```

Out[37]:
```
0                  Afghanistan
3                       Angola
17                       Benin
21                    Botswana
25                Burkina Faso
26                     Burundi
28                    Cameroon
31    Central African Republic
32                        Chad
36                     Comoros
37            Congo, Dem. Rep.
38                 Congo, Rep.
40               Cote d'Ivoire
49           Equatorial Guinea
50                     Eritrea
55                       Gabon
56                      Gambia
59                       Ghana
63                      Guinea
64               Guinea-Bissau
66                       Haiti
72                        Iraq
80                       Kenya
81                    Kiribati
84                         Lao
87                     Lesotho
88                     Liberia
93                  Madagascar
```

```
94                       Malawi
97                         Mali
99                   Mauritania
106                  Mozambique
108                     Namibia
112                       Niger
113                     Nigeria
116                    Pakistan
126                      Rwanda
129                     Senegal
132                Sierra Leone
136             Solomon Islands
137                South Africa
142                       Sudan
147                    Tanzania
149                 Timor-Leste
150                        Togo
155                      Uganda
165                       Yemen
166                      Zambia
Name: country, dtype: object
```

In [38]:

```
top_kmeans = df_km1[df_km1['cluster_id']==1].sort_values(by=["child_mort","gdpp","in
top_kmeans = top_kmeans.reset_index().drop('index',axis=1)
top_kmeans.head(10)
```

Out[38]:

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

In [39]:

```
top_10 = top_kmeans.iloc[:10]
top_10['country'].reset_index().drop('index',axis = 1)
```

Out[39]:

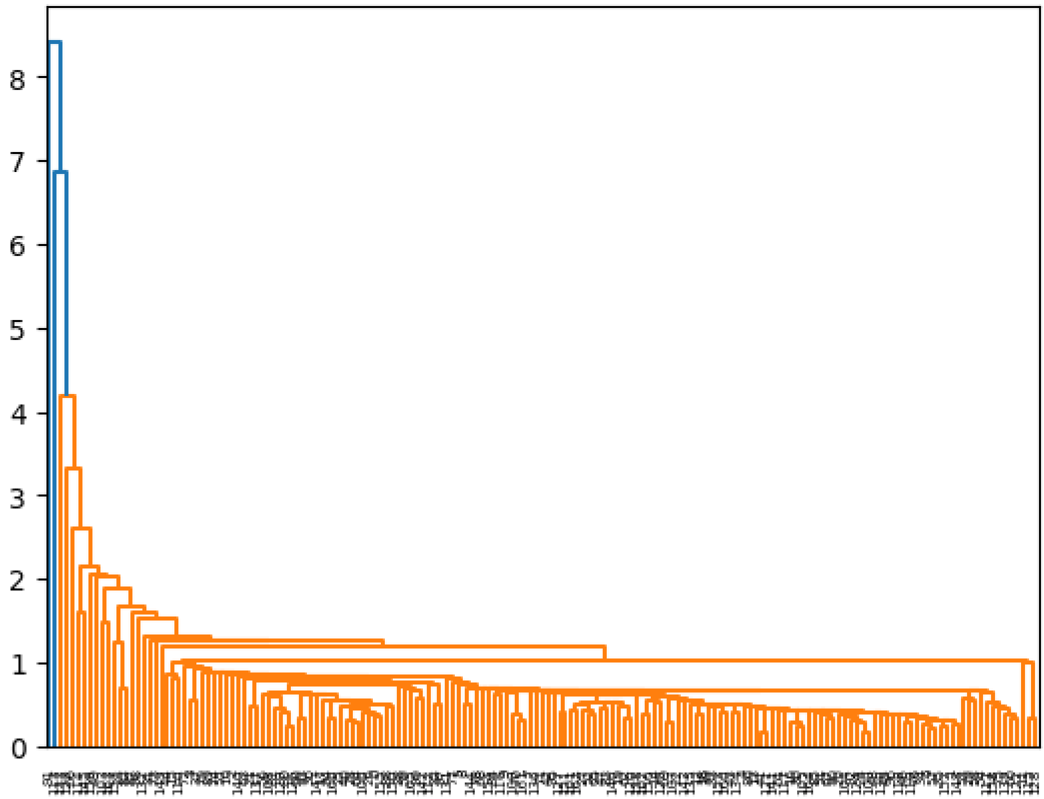
	country
0	Haiti
1	Sierra Leone
2	Chad

	country
3	Central African Republic
4	Mali
5	Nigeria
6	Niger
7	Angola
8	Congo, Dem. Rep.
9	Burkina Faso

Even with cluster 1, the 10 countries mentioned have the worst performance across all three considered metrics.

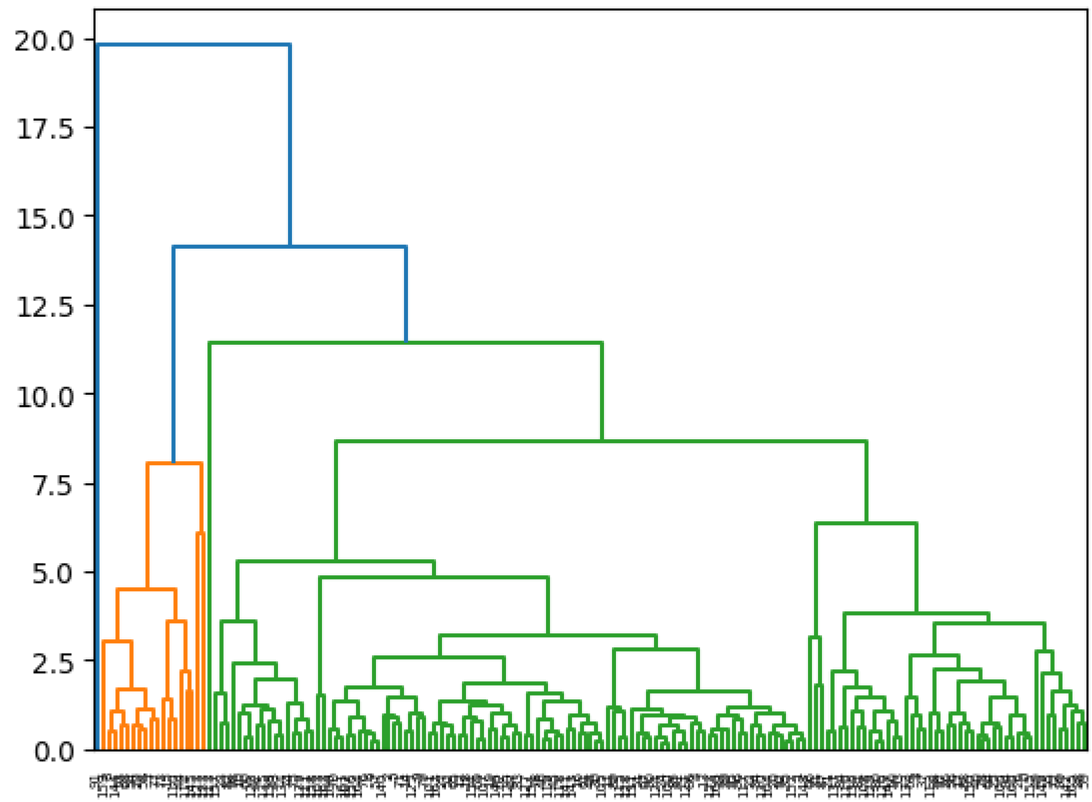
Hierarchical

In [40]:
```
#single linkage.
mergings_single = linkage(dfx, method = "single", metric = 'euclidean')
dendrogram(mergings_single)
plt.show()
```



Unsatisfactory performance with single linkage, so complete will be attempted.

In [41]:
```
mergings_complete = linkage(dfx, method = "complete", metric = 'euclidean')
dendrogram(mergings_complete)
plt.show()
```



In [42]:
```
cluster_labels = cut_tree(mergings_complete, n_clusters=3).reshape(-1, )
cluster_labels
```

Out[42]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 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,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 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])
```

In [43]:
```
df_hm = pd.concat([df, pd.Series(cluster_labels)], axis = 1)
df_hm.columns = ['country','child_mort','exports','health','imports','income','infla
df_hm.head()
```

Out[43]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdp
0	Afghanistan	90.2	55.30	41.9174	248.297	1610	9.44	56.2	5.82	55
1	Albania	16.6	1145.20	267.8950	1987.740	9930	4.49	76.3	1.65	409
2	Algeria	27.3	1712.64	185.9820	1400.440	12900	16.10	76.5	2.89	446
3	Angola	119.0	2199.19	100.6050	1514.370	5900	22.40	60.1	6.16	353
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100	1.44	76.8	2.13	1220

In [44]:
```
df_hm['cluster_labels'].value_counts()
```

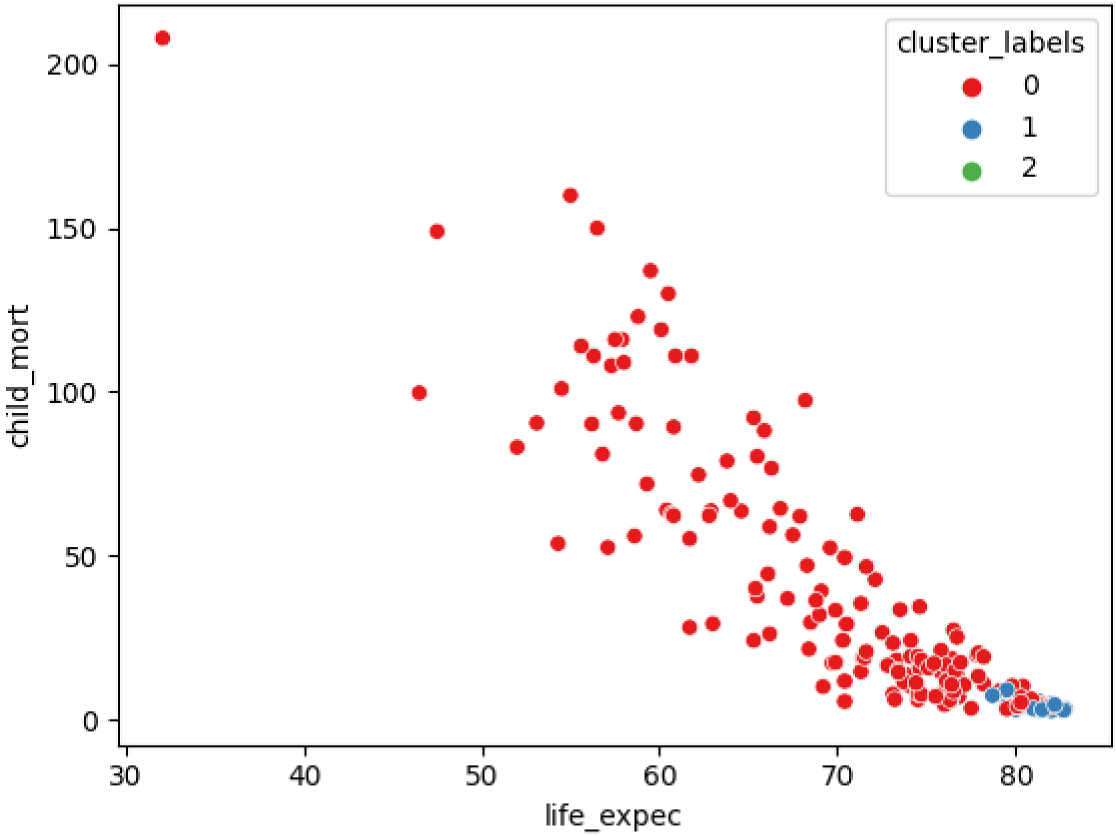
Out[44]:
```
0    148
1     18
```

```
2      1
Name: cluster_labels, dtype: int64
```

Cluster Visualisation

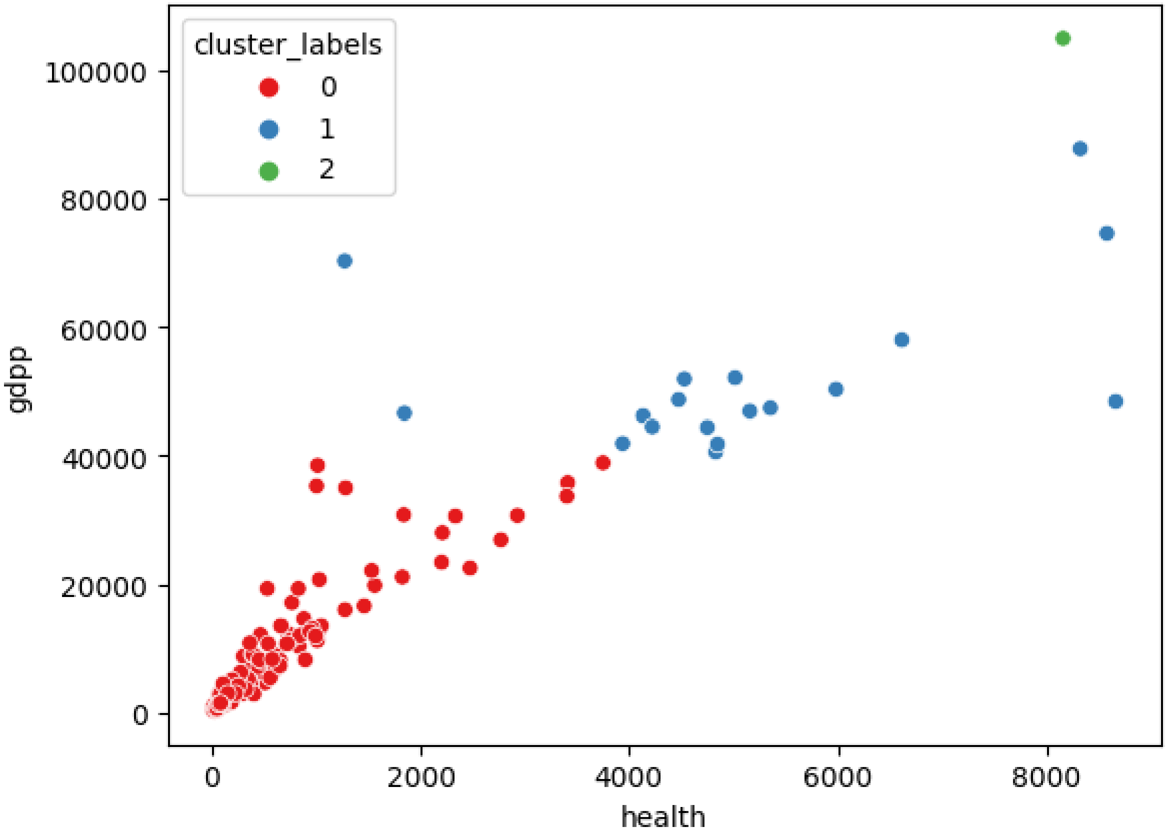
In [45]:

```
sns.scatterplot(x = 'life_expec', y = 'child_mort', hue ='cluster_labels', legend =
plt.show()
```



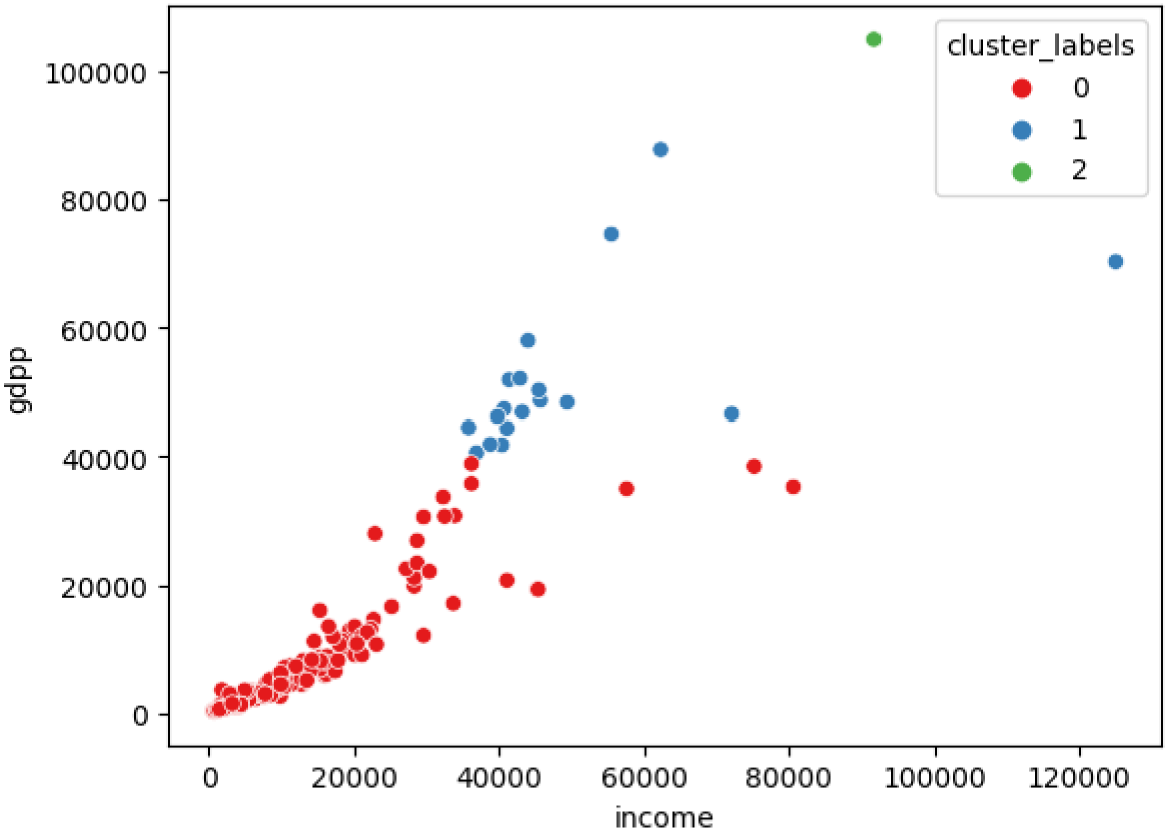
In [46]:

```
sns.scatterplot(x = 'health', y = 'gdpp', hue ='cluster_labels', legend = 'full', da
plt.show()
```



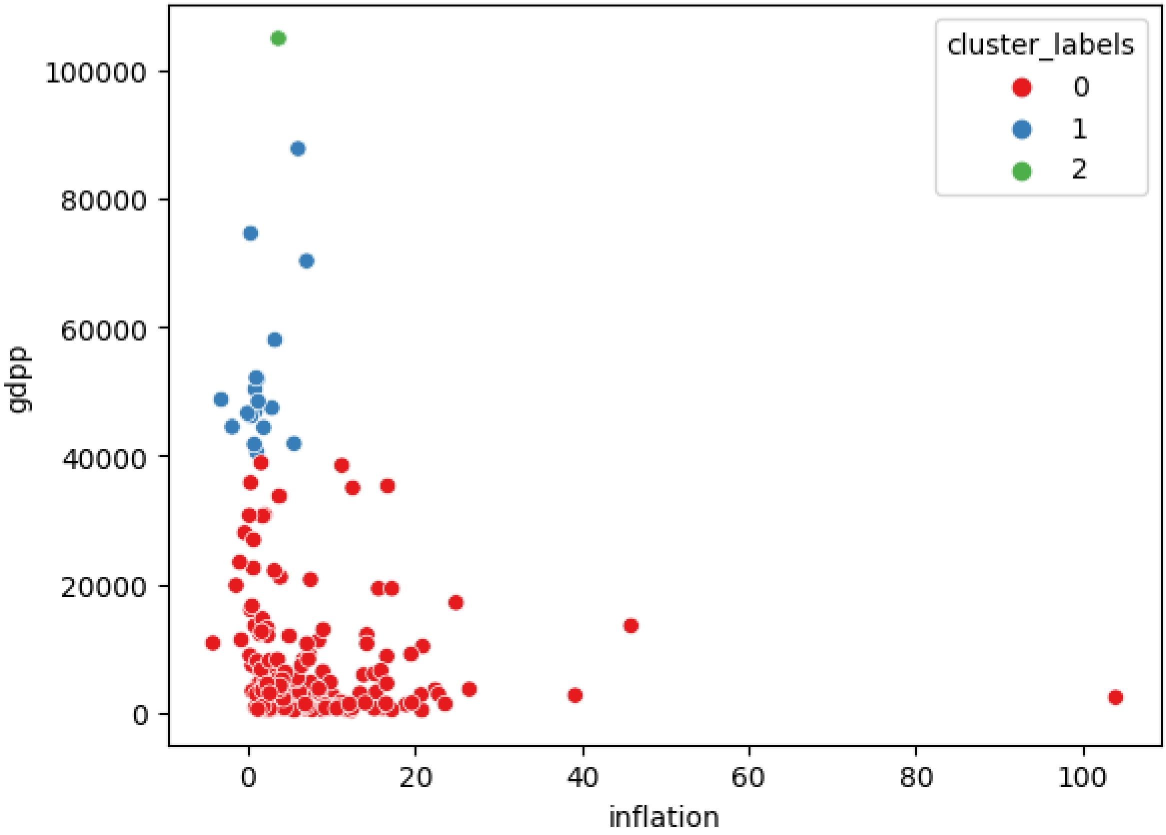
In [47]:

```
sns.scatterplot(x = 'income', y = 'gdpp', hue ='cluster_labels', legend = 'full', da
plt.show()
```

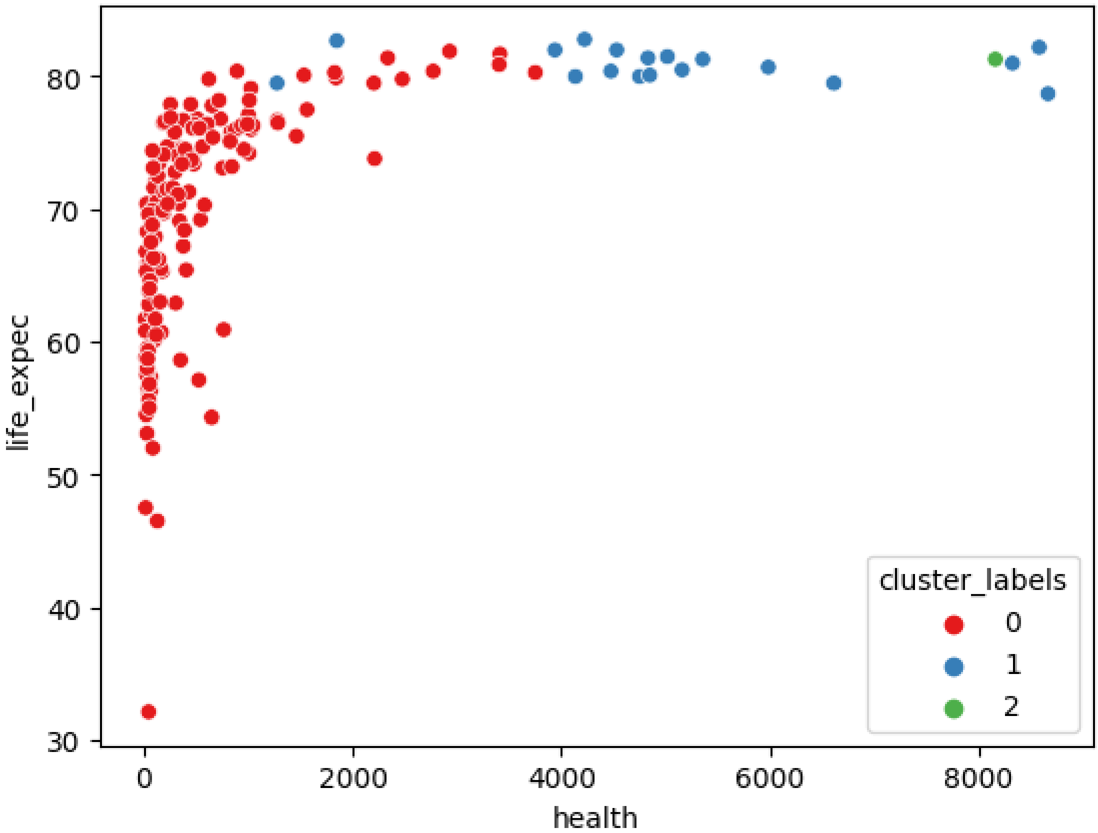


In [48]:

```
sns.scatterplot(x = 'inflation', y = 'gdpp', hue ='cluster_labels', legend = 'full',
plt.show()
```

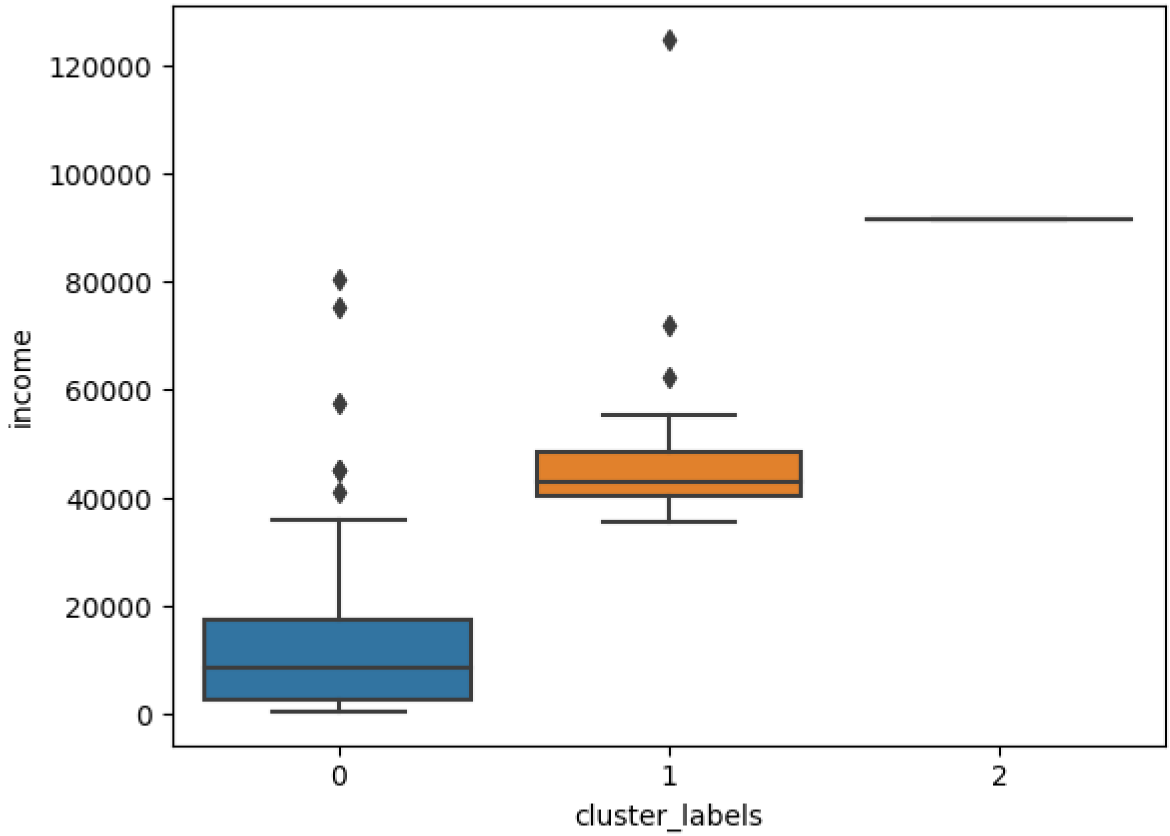


In [49]:
```
sns.scatterplot(x = 'health', y = 'life_expec', hue ='cluster_labels', legend = 'ful
plt.show()
```



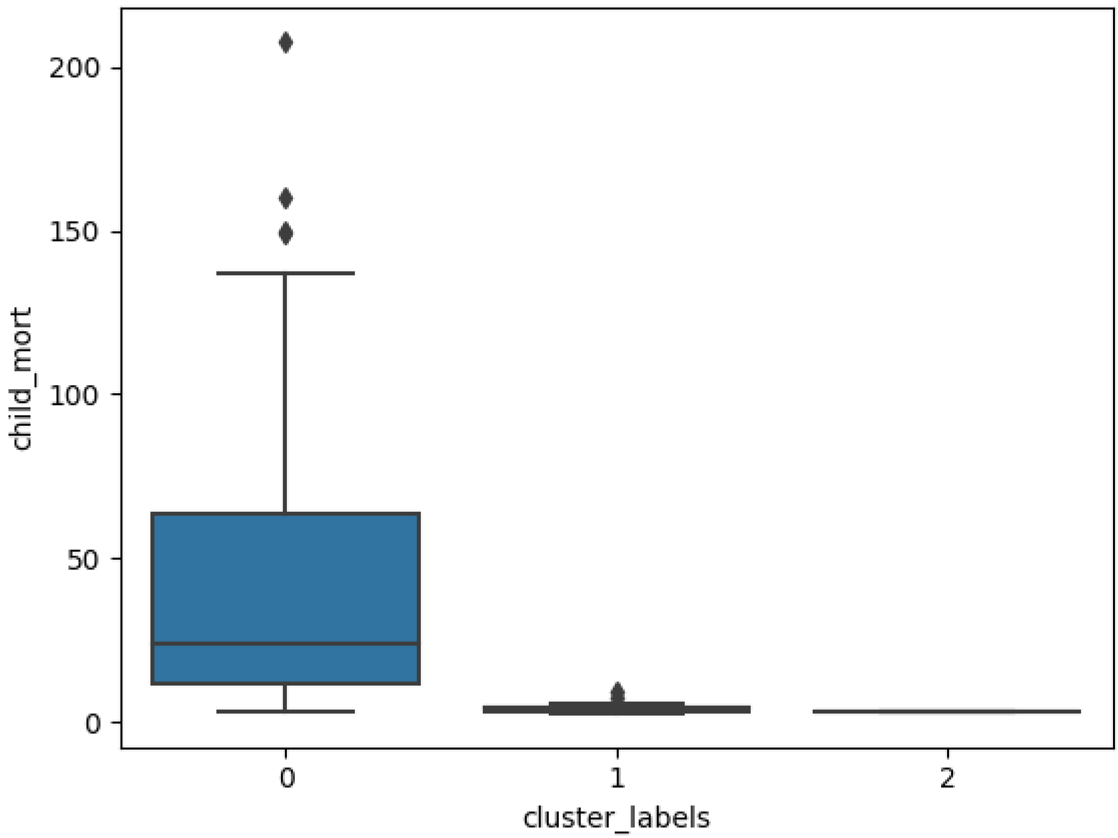
Using boxplots for checking the lowest performaning cluster.

In [50]:
```
sns.boxplot(data = df_hm, x = 'cluster_labels',y='income')
plt.show()
```



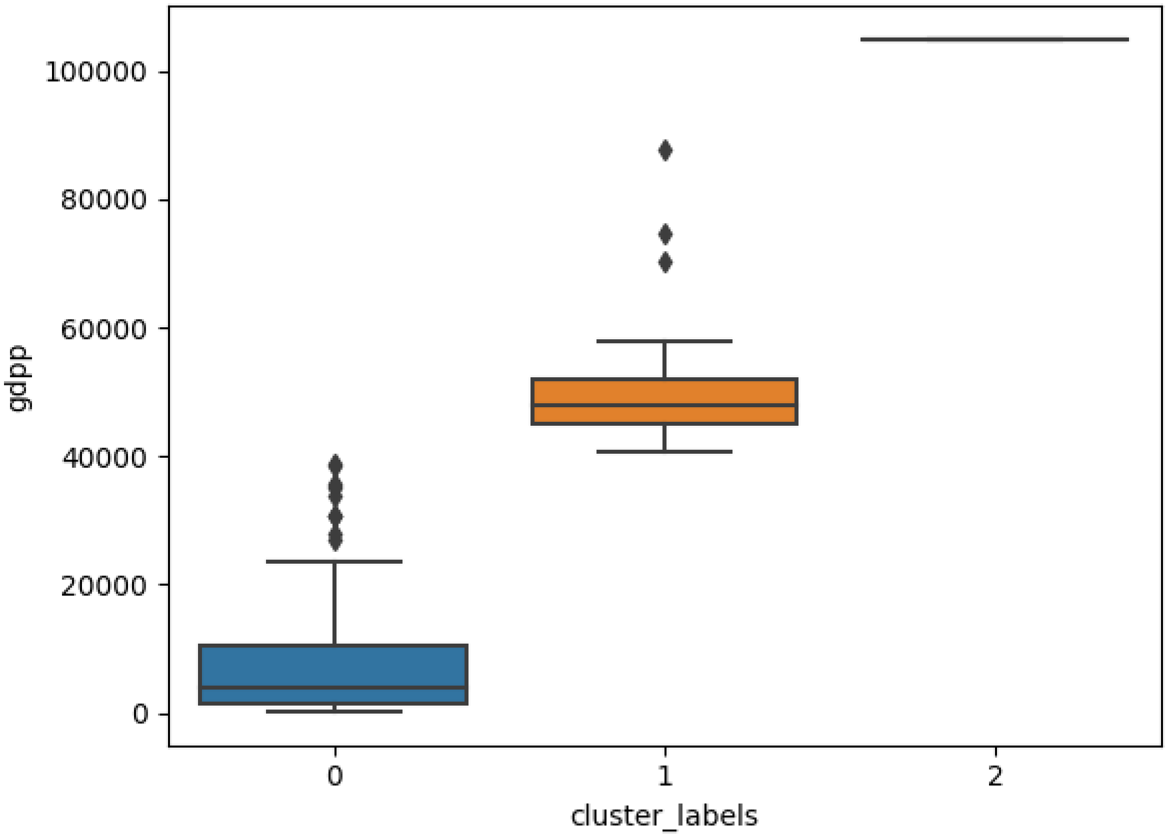
In [51]:

```
sns.boxplot(data = df_hm, x = 'cluster_labels',y = 'child_mort')
plt.show()
```



In [52]:

```
sns.boxplot(data = df_hm,x = 'cluster_labels', y = 'gdpp')
plt.show()
```



Obviously, cluster 0 has the worst performance across the metrics.

In [53]:
```
df_hm[df_hm['cluster_labels'] == 0]['country']
```

Out[53]:
```
0              Afghanistan
1                  Albania
2                  Algeria
3                   Angola
4      Antigua and Barbuda
              ...
162                Vanuatu
163              Venezuela
164                Vietnam
165                  Yemen
166                 Zambia
Name: country, Length: 148, dtype: object
```

In [54]:
```
top_h = df_hm[df_hm['cluster_labels']==0].sort_values(by=["child_mort","gdpp","incom
top_h = top_h.reset_index().drop('index',1)
top_h.head(10)
```

```
<ipython-input-54-9a473548feca>:2: FutureWarning: In a future version of pandas all a
rguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
  top_h = top_h.reset_index().drop('index',1)
```

Out[54]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Haiti	208.0	101.286	45.7442	428.314	1500	5.45	32.1	3.33	662
1	Sierra Leone	160.0	67.032	52.2690	137.655	1220	17.20	55.0	5.20	399
2	Chad	150.0	330.096	40.6341	390.195	1930	6.39	56.5	6.59	897
3	Central African Republic	149.0	52.628	17.7508	118.190	888	2.01	47.5	5.21	446

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
4	Mali	137.0	161.424	35.2584	248.508	1870	4.37	59.5	6.55	708
5	Nigeria	130.0	589.490	118.1310	405.420	5150	104.00	60.5	5.84	2330
6	Niger	123.0	77.256	17.9568	170.868	814	2.55	58.8	7.49	348
7	Angola	119.0	2199.190	100.6050	1514.370	5900	22.40	60.1	6.16	3530
8	Congo, Dem. Rep.	116.0	137.274	26.4194	165.664	609	20.80	57.5	6.54	334
9	Burkina Faso	116.0	110.400	38.7550	170.200	1430	6.81	57.9	5.87	575

In [55]:

```
top_10 = top_h.iloc[:10]
top_10['country'].reset_index().drop('index',axis = 1)
```

Out[55]:

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