finding-countries-in-need-of-financial-aid

June 29, 2022

[4]:
```
#custom imports

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans
from sklearn.cluster import MeanShift
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors
from sklearn.cluster import MeanShift
import plotly.express as px
from sklearn.metrics import silhouette_score
from sklearn.cluster import AgglomerativeClustering

import plotly.graph_objects as go
from plotly.subplots import make_subplots
import folium
```

In this project we will assist HELP International to decide which country requires the financial support based on socio-economic status of the country. We will use some unsupervised learning algorithms to find the right countires for the need of aid.

Let's understand the columns provided in this dataset.

Country - Name of the Country **child_mort** - Death of children under 5 years of age per 1000 live births **exports** - Exports of goods and services per capita. Given as percentage of the GDP per capita **health** - Total health spending per capita. Given as percentage of GDP per capita **imports** - Imports of goods and services per capita. Given as percentage of the GDP per capita **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 rates remain the same. **gdpp** - The GDP per capita. Calculated as the Total GDP divided by the total population.

[5]:
```
#get the dataframe
df = pd.read_csv("/../Country-data.csv")
```

[6]:
```
#overview of the data
df.head()
```

[6]:
```
               country  child_mort  exports  health  imports  income  \
0          Afghanistan        90.2     10.0    7.58     44.9    1610
1              Albania        16.6     28.0    6.55     48.6    9930
2              Algeria        27.3     38.4    4.17     31.4   12900
3               Angola       119.0     62.3    2.85     42.9    5900
4  Antigua and Barbuda        10.3     45.5    6.03     58.9   19100

   inflation  life_expec  total_fer   gdpp
0       9.44        56.2       5.82    553
1       4.49        76.3       1.65   4090
2      16.10        76.5       2.89   4460
3      22.40        60.1       6.16   3530
4       1.44        76.8       2.13  12200
```

0.1 Data Cleaning and Data Wrangling

Firstly, we will check for the datatypes of each column.

[7]:
```
df.dtypes
```

[7]:
```
country        object
child_mort    float64
exports       float64
health        float64
imports       float64
income          int64
inflation     float64
life_expec    float64
total_fer     float64
gdpp            int64
dtype: object
```

[8]:
```
df.shape
```

[8]:
```
(167, 10)
```

All the data types seems to be correct for that feature.

Secondly, we will check for missing or null values.

[9]:
```
df.isnull().value_counts()
```

[9]:
```
country  child_mort  exports  health  imports  income  inflation  life_expec
total_fer  gdpp
False    False       False    False   False    False   False      False
```

```
False      False    167
dtype: int64
```

There seems to be no missing values in this dataset.

1 Exploratory Data Analysis

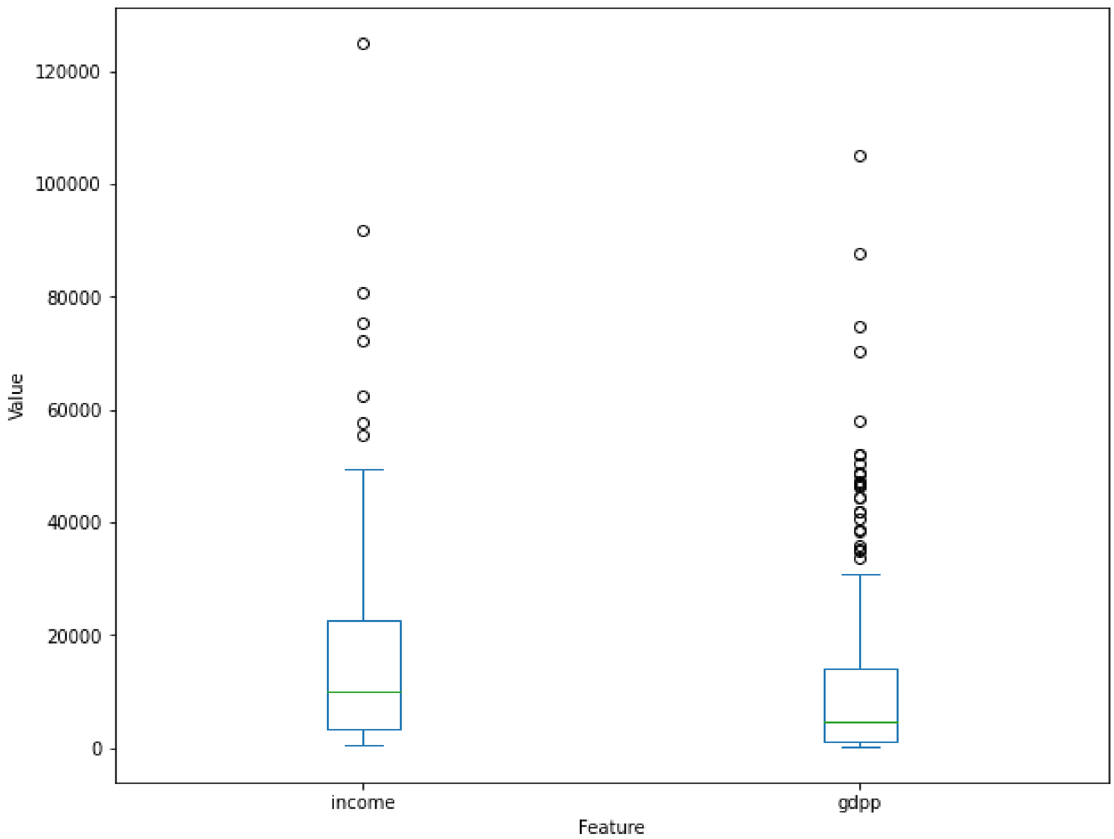
[10]:
```
df.describe()
```

[10]:
```
        child_mort     exports      health     imports         income  \
count   167.000000  167.000000  167.000000  167.000000     167.000000
mean     38.270060   41.108976    6.815689   46.890215   17144.688623
std      40.328931   27.412010    2.746837   24.209589   19278.067698
min       2.600000    0.109000    1.810000    0.065900     609.000000
25%       8.250000   23.800000    4.920000   30.200000    3355.000000
50%      19.300000   35.000000    6.320000   43.300000    9960.000000
75%      62.100000   51.350000    8.600000   58.750000   22800.000000
max     208.000000  200.000000   17.900000  174.000000  125000.000000

         inflation  life_expec   total_fer           gdpp
count   167.000000  167.000000  167.000000     167.000000
mean      7.781832   70.555689    2.947964   12964.155689
std      10.570704    8.893172    1.513848   18328.704809
min      -4.210000   32.100000    1.150000     231.000000
25%       1.810000   65.300000    1.795000    1330.000000
50%       5.390000   73.100000    2.410000    4660.000000
75%      10.750000   76.800000    3.880000   14050.000000
max     104.000000   82.800000    7.490000  105000.000000
```

Let's try to visualize the mean, standard deviation and interquartile range to get better idea of the spread of the data. We will separate GDPP and Net income from the other features for visualization because as seen from Max values there is huge difference between each feature.

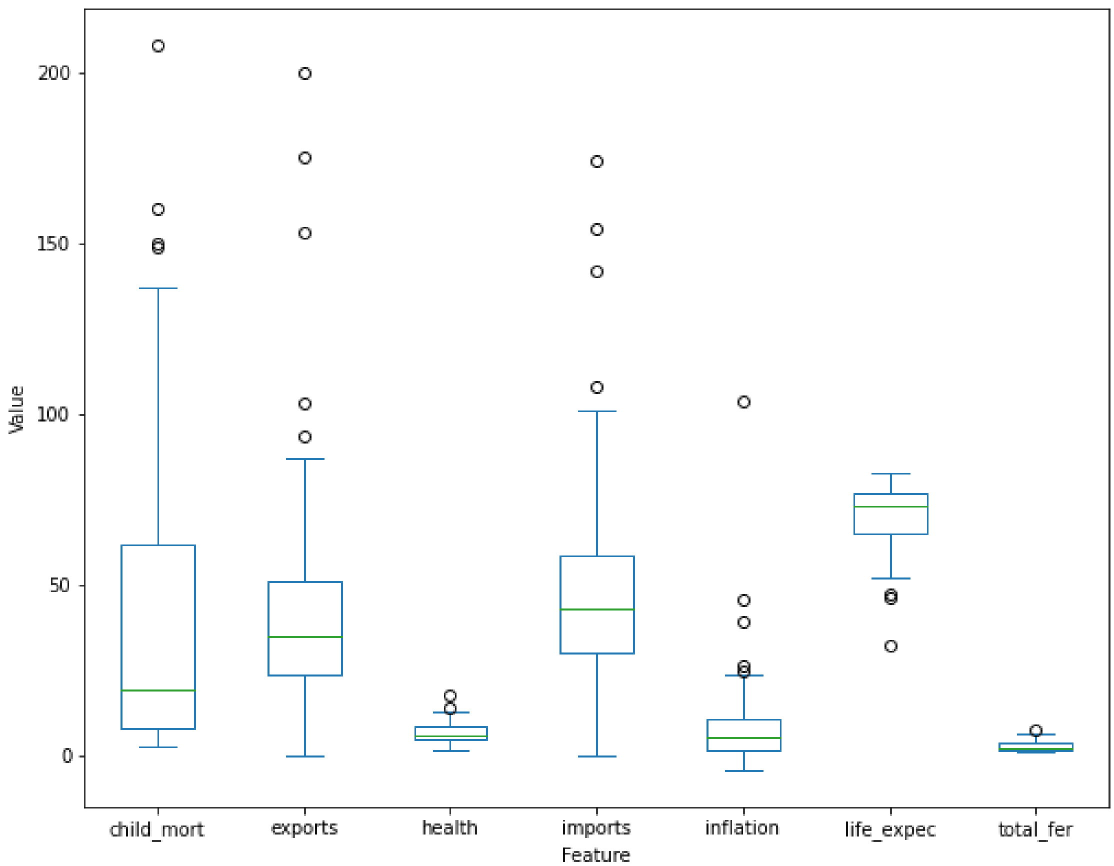
[11]:
```
#fig,ax = plt.subplots(figsize=(10,8))
df[['income','gdpp']].plot(kind='box', figsize=(10,8))
plt.xlabel("Feature")
plt.ylabel("Value")
```

[11]:
```
Text(0, 0.5, 'Value')
```




```
[12]: df[['child_mort','exports','health','imports','inflation','life_expec','total_fer']].
       ↪plot(kind='box', figsize=(10,8))
      plt.xlabel("Feature")
      plt.ylabel("Value")
```

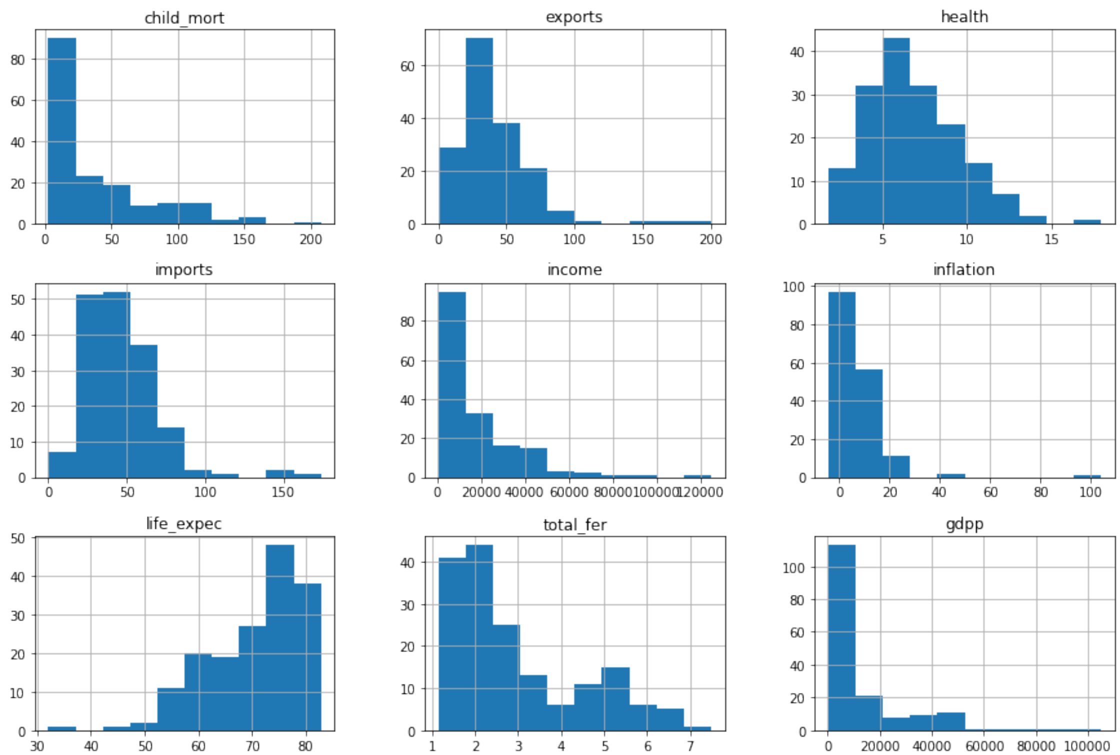
```
[12]: Text(0, 0.5, 'Value')
```



There are few outliers for all these features for some unknown reasons. It needs more in-depth knowledge to deal with them. Plotting histograms will provide more insights.

[13]:
```
df.hist(figsize=(15,10))
```

[13]:
```
array([[<AxesSubplot:title={'center':'child_mort'}>,
        <AxesSubplot:title={'center':'exports'}>,
        <AxesSubplot:title={'center':'health'}>],
       [<AxesSubplot:title={'center':'imports'}>,
        <AxesSubplot:title={'center':'income'}>,
        <AxesSubplot:title={'center':'inflation'}>],
       [<AxesSubplot:title={'center':'life_expec'}>,
        <AxesSubplot:title={'center':'total_fer'}>,
        <AxesSubplot:title={'center':'gdpp'}>]], dtype=object)
```



All of these features appears to be highly skewed. Distance Based Spatial Clustering of Applications With Noise, inshort DBSCAN, can be used here which specifically leave out the outliers from clustering the data. We will use DBSCAN algorithm for clustering initially and then we will try to compare the accuracy with other clustering methods like K-means and Mean Shift.

1.1 Scaling data

Scaling the data is important as the values of different features show a high range of differences and this could lead to errors in the distacne calculation. By generalizing the data points the distance can be lowered between them thus bringing them to similar level.

[14]:
```
#removing the country column from the main dataset
df_final = df.iloc[:, 1:]
df_final
```

[14]:
```
     child_mort  exports  health  imports  income  inflation  life_expec  \
0          90.2     10.0    7.58     44.9    1610       9.44        56.2
1          16.6     28.0    6.55     48.6    9930       4.49        76.3
2          27.3     38.4    4.17     31.4   12900      16.10        76.5
3         119.0     62.3    2.85     42.9    5900      22.40        60.1
4          10.3     45.5    6.03     58.9   19100       1.44        76.8
..          …        …       …        …        …          …
162        29.2     46.6    5.25     52.7    2950       2.62        63.0
163        17.1     28.5    4.91     17.6   16500      45.90        75.4
```

```
164         23.3     72.0    6.84     80.2    4490      12.10          73.1
165         56.3     30.0    5.18     34.4    4480      23.60          67.5
166         83.1     37.0    5.89     30.9    3280      14.00          52.0

     total_fer   gdpp
0         5.82    553
1         1.65   4090
2         2.89   4460
3         6.16   3530
4         2.13  12200
..         ...    ...
162       3.50   2970
163       2.47  13500
164       1.95   1310
165       4.67   1310
166       5.40   1460

[167 rows x 9 columns]
```

[15]:

```python
#Creating an object of StandardScaler
scaled = StandardScaler()

#fitting and tranforming the data to a new dataframe
df_scaled = pd.DataFrame(scaled.fit_transform(df_final), columns =df_final.
 ↪columns)
df_scaled
```

[15]:

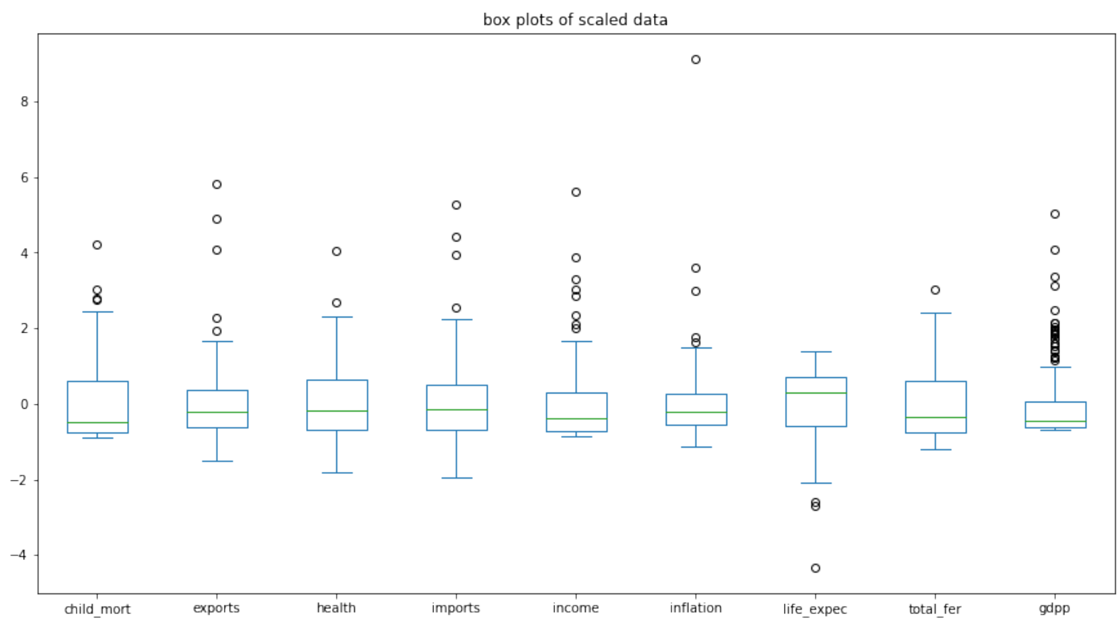
```
     child_mort   exports    health   imports    income  inflation  \
0      1.291532 -1.138280  0.279088 -0.082455 -0.808245   0.157336
1     -0.538949 -0.479658 -0.097016  0.070837 -0.375369  -0.312347
2     -0.272833 -0.099122 -0.966073 -0.641762 -0.220844   0.789274
3      2.007808  0.775381 -1.448071 -0.165315 -0.585043   1.387054
4     -0.695634  0.160668 -0.286894  0.497568  0.101732  -0.601749
..          ...       ...       ...       ...       ...        ...
162   -0.225578  0.200917 -0.571711  0.240700 -0.738527  -0.489784
163   -0.526514 -0.461363 -0.695862 -1.213499 -0.033542   3.616865
164   -0.372315  1.130305  0.008877  1.380030 -0.658404   0.409732
165    0.448417 -0.406478 -0.597272 -0.517472 -0.658924   1.500916
166    1.114951 -0.150348 -0.338015 -0.662477 -0.721358   0.590015

     life_expec  total_fer      gdpp
0     -1.619092   1.902882 -0.679180
1      0.647866  -0.859973 -0.485623
2      0.670423  -0.038404 -0.465376
3     -1.179234   2.128151 -0.516268
4      0.704258  -0.541946 -0.041817
..          ...        ...       ...
```

```
162  -0.852161   0.365754 -0.546913
163   0.546361  -0.316678  0.029323
164   0.286958  -0.661206 -0.637754
165  -0.344633   1.140944 -0.637754
166  -2.092785   1.624609 -0.629546

[167 rows x 9 columns]
```

Plotting the box plots again to visualize the scaled data distrbution.

[16]:
```
df_scaled.plot(kind='box', figsize=(15,8), title="box plots of scaled data")
```

[16]:
```
<AxesSubplot:title={'center':'box plots of scaled data'}>
```



The data is nicely scaled to bring the values of all columns to a comparable smaller range.

Before we proceed with DBSCAN, there are two main important parameters that needs to be decided or estimated. One is Epsilon which is a considered as a radius that will cover the nearest point to expand the cluster. Second is min_samples which is the mininum number of samples in that radius of epsilon.

To find right epsilon we will use NearestNeighbors algorithm from sklearn.

[17]:
```
#creating an object for NearestNeighbors
nbrs = NearestNeighbors(n_neighbors =4)  # for k=4, it will find 3 nearest␣
↪neighbors

#getting the distances and indices by fitting the scaled data to the model nbrs
```

```
distances, indices = nbrs.fit(df_scaled).kneighbors(df_scaled)

#sorting the distance values in ascending order
distances = np.sort(distances, axis=0)

#filter the distance to not have first column of zeros have
#distances = distances[:,1:]
```

[18]:
```
#plotting the line plot of distances
px.line(distances, title = "Nearest Neighbor distance values for each Index")
```

These four lines indicates the column number for the "distances" array and as the first column is 0, we see a blue line corresoinding to that in the above plot. Higher values of distances indicates heavy outliers as the distance from the core point to neighbor points increases.

Now discussing about where the curve makes an elbow here,it can be inferred that for all the curved lines elbow starts to form around at 1. If we consider the red line then the value can be estimated to be around 1.2-1.4 and specifically 1.24. So we will use an epsilon of 1.24 and minimum samples per radius of epsilon to be 4.

1.2 DBSCAN algorithm

[19]:
```
#create an object for DBSCAN
classify_db = DBSCAN(eps=1.24, min_samples=3)

#fit the data to the object
classify_db.fit(df_scaled)

#generate the labels
labels = classify_db.labels_
```

```
#add the labels to the original main dataframe
df['class'] = list(labels)

#check the silhouette score
score = silhouette_score(df_scaled, labels)

print(f'Average silhouette score is {score}')
```

```
Average silhouette score is 0.13554366371098675
```

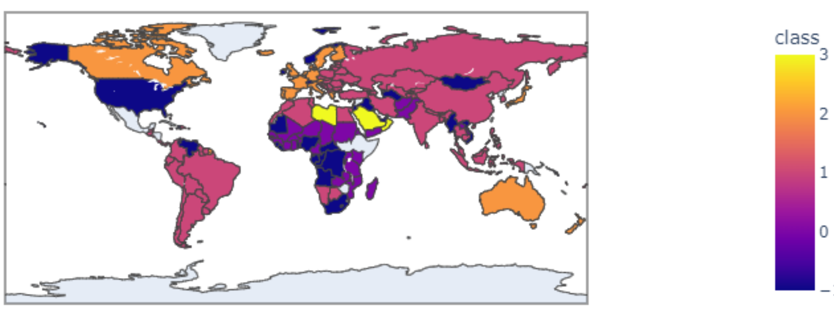
The average silhouette score is very low and indicates overlapping clusters with DBSCAN method. We need to compare this with other clustering algorithms to find the best one.

[20]:
```
df.groupby('class')['country'].count()
```

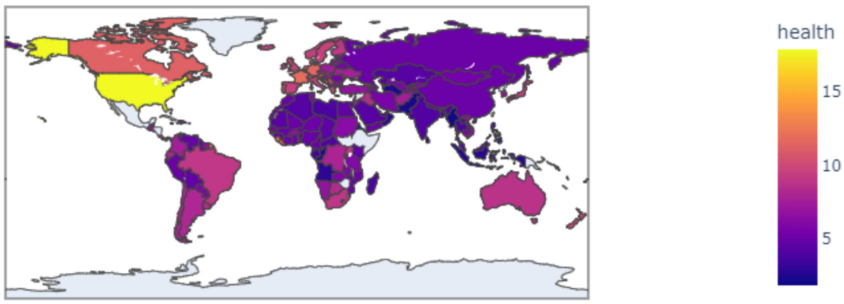
[20]:
```
class
-1    39
 0    27
 1    78
 2    19
 3     4
Name: country, dtype: int64
```

Let's visualize the countries classified as per the labeling done in DBSCAN method. For this we will use plotly and choropleth libraries to map the countries.

[21]:
```
px.choropleth(df, locationmode='country names',locations='country',␣
↪color='class',
                           color_discrete_map = {'-1':'red', '0':'blue',
                                  '1':'yellow', '2':'green'},
                           labels={'unemp':'unemployment rate'}
                          )
```



[22]:
```
px.choropleth(df, locationmode='country names',locations='country',␣
↪color='health',

              )
```



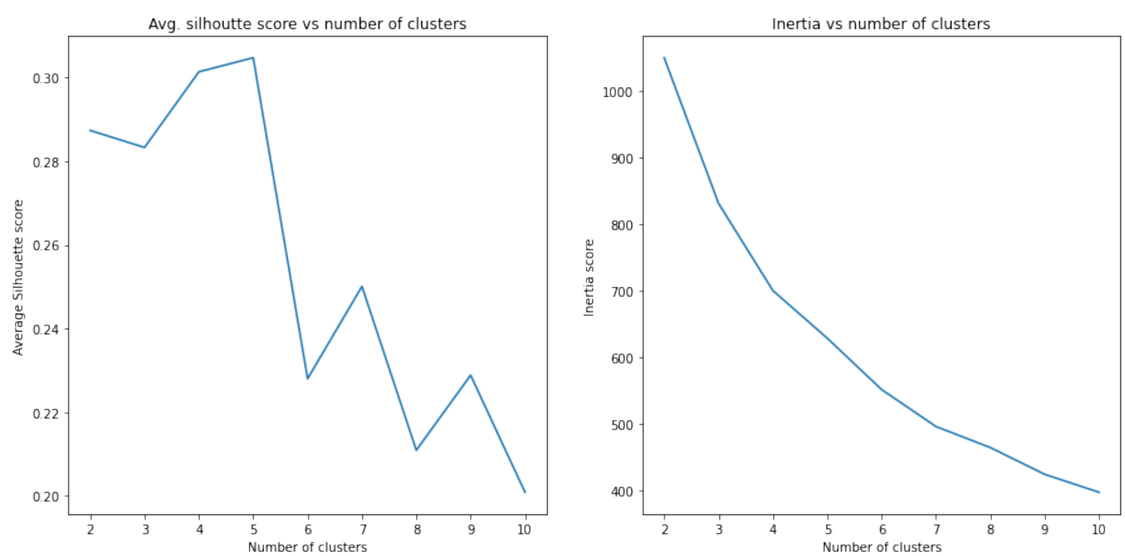
1.3 K Means Clustering

Let's also confirm the value of clusters using the elbow method

[23]:
```
'''
A function that will run the Kmeans classifier and provide labels.
The labels will be used as classes column for the dataframe and
function will also return average silhouette score. Inertia is also
calculated and returned as a list for number of clusters.

'''
def kmeans(n):
    #create a Kmeans object
    classify_km = KMeans(n_clusters=n, random_state=24)

    #fit the data to the model
    classify_km.fit(df_scaled)

    #get the labels
    KM_labels = classify_km.labels_

    df.drop('class', axis=1, inplace=True)
    df['class'] = list(KM_labels)

    #check the silhouette score
    score = silhouette_score(df_scaled, KM_labels)
```

```
    inertia = classify_km.inertia_
    return score, inertia

#create an empty list to store average silhouette sore values for kmeans
sill_score = []
inertia =[]
#loop through the range of k vakues to get its respective score
for i in range(2,11):
    sill_score.append(kmeans(i)[0])
    inertia.append(kmeans(i)[1])


#plot the line graph for silhouette score
fig, ax = plt.subplots(1,2,figsize=(15,7))
ax[0].plot(range(2,11),sill_score)
ax[0].set_xlabel("Number of clusters")
ax[0].set_ylabel("Average Silhouette score")
ax[0].set_title("Avg. silhoutte score vs number of clusters")

#plot the line graph for inertia values
ax[1].plot(range(2,11),inertia)
ax[1].set_xlabel("Number of clusters")
ax[1].set_ylabel("Inertia score")
ax[1].set_title("Inertia vs number of clusters")
```

[23]: Text(0.5, 1.0, 'Inertia vs number of clusters')

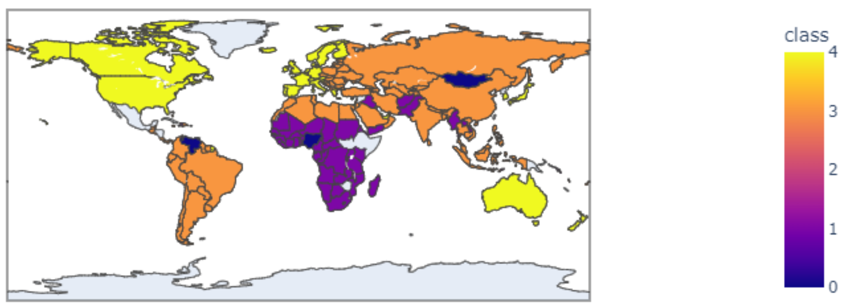


Average Silhouette score is highest 0.3 at value of 5 clusters. So we will take 5 clusters for K means algorithm. The elbow curve is not useful visually but it can be said that the silhouette score maxes as 5 clusters and the elbow occurs at 5 as well.

[24]:
```python
score = kmeans(5)
print(f'Average silhoutte score for K means is {score}')
```

```
Average silhoutte score for K means is (0.30475221266676467, 628.8066422564427)
```

[25]:
```python
#plotting the classes and countries
px.choropleth(df, locationmode='country names',locations='country',␣
↪color='class',
                            color_continuous_scale="Plasma",
                            labels={'unemp':'unemployment rate'}
                           )
```



[26]:
```python
#assigning labels column to scaled dataframe
df_scaled['class'] = df['class']

#creating a dataframe that has only mean values of each feature for all classes
polar= df_scaled.groupby('class').mean().reset_index()
print(polar)

polar = pd.melt(polar,id_vars=["class"])
polar
```

```
   class  child_mort   exports    health   imports    income  inflation  \
0      0    0.484065 -0.278413 -0.611878 -0.676287 -0.382826   5.242572
1      1    1.292620 -0.441377 -0.163124 -0.170610 -0.690788   0.200143
2      2   -0.849003  4.935673 -0.008163  4.548058  2.439542  -0.504206
3      3   -0.434852  0.024526 -0.193803  0.065845 -0.203380  -0.114173
4      4   -0.828609  0.172621  0.859190 -0.296373  1.462275  -0.478189

   life_expec  total_fer      gdpp
```

0	-0.359671	0.465138	-0.372346
1	-1.261473	1.306997	-0.606493
2	1.226824	-1.038863	2.440797
3	0.297828	-0.459087	-0.324708
4	1.107649	-0.763681	1.661902

[26]:

	class	variable	value
0	0	child_mort	0.484065
1	1	child_mort	1.292620
2	2	child_mort	-0.849003
3	3	child_mort	-0.434852
4	4	child_mort	-0.828609
5	0	exports	-0.278413
6	1	exports	-0.441377
7	2	exports	4.935673
8	3	exports	0.024526
9	4	exports	0.172621
10	0	health	-0.611878
11	1	health	-0.163124
12	2	health	-0.008163
13	3	health	-0.193803
14	4	health	0.859190
15	0	imports	-0.676287
16	1	imports	-0.170610
17	2	imports	4.548058
18	3	imports	0.065845
19	4	imports	-0.296373
20	0	income	-0.382826
21	1	income	-0.690788
22	2	income	2.439542
23	3	income	-0.203380
24	4	income	1.462275
25	0	inflation	5.242572
26	1	inflation	0.200143
27	2	inflation	-0.504206
28	3	inflation	-0.114173
29	4	inflation	-0.478189
30	0	life_expec	-0.359671
31	1	life_expec	-1.261473
32	2	life_expec	1.226824
33	3	life_expec	0.297828
34	4	life_expec	1.107649
35	0	total_fer	0.465138
36	1	total_fer	1.306997
37	2	total_fer	-1.038863
38	3	total_fer	-0.459087
39	4	total_fer	-0.763681

```
40      0        gdpp -0.372346
41      1        gdpp -0.606493
42      2        gdpp  2.440797
43      3        gdpp -0.324708
44      4        gdpp  1.661902
```

[27]:
```python
#checking the categories and counts of countries
df_scaled['class'].value_counts()
```

[27]:
```
3    83
1    48
4    30
2     3
0     3
Name: class, dtype: int64
```

[45]:
```python
%%capture
fig = px.line_polar(polar, r="value", theta="variable", color="class",␣
 ↪line_close=True, template="plotly_dark")
```

[46]:
```python
fig.show()
```

Let's try to divide the countries according to the labels.

[29]:
```python
df[df['class']==4]
```

[29]:
```
               country  child_mort  exports  health  imports  income  \
7            Australia         4.8     19.8    8.73     20.9   41400
8              Austria         4.3     51.3   11.00     47.8   43200
15             Belgium         4.5     76.4   10.70     74.7   41100
23              Brunei        10.5     67.4    2.84     28.0   80600
```

29	Canada	5.6	29.1	11.30	31.0	40700
42	Cyprus	3.6	50.2	5.97	57.5	33900
44	Denmark	4.1	50.5	11.40	43.6	44000
53	Finland	3.0	38.7	8.95	37.4	39800
54	France	4.2	26.8	11.90	28.1	36900
58	Germany	4.2	42.3	11.60	37.1	40400
60	Greece	3.9	22.1	10.30	30.7	28700
68	Iceland	2.6	53.4	9.40	43.3	38800
73	Ireland	4.2	103.0	9.19	86.5	45700
74	Israel	4.6	35.0	7.63	32.9	29600
75	Italy	4.0	25.2	9.53	27.2	36200
77	Japan	3.2	15.0	9.49	13.6	35800
82	Kuwait	10.8	66.7	2.63	30.4	75200
110	Netherlands	4.5	72.0	11.90	63.6	45500
111	New Zealand	6.2	30.3	10.10	28.0	32300
114	Norway	3.2	39.7	9.48	28.5	62300
122	Portugal	3.9	29.9	11.00	37.4	27200
123	Qatar	9.0	62.3	1.81	23.8	125000
135	Slovenia	3.2	64.3	9.41	62.9	28700
138	South Korea	4.1	49.4	6.93	46.2	30400
139	Spain	3.8	25.5	9.54	26.8	32500
144	Sweden	3.0	46.2	9.63	40.7	42900
145	Switzerland	4.5	64.0	11.50	53.3	55500
157	United Arab Emirates	8.6	77.7	3.66	63.6	57600
158	United Kingdom	5.2	28.2	9.64	30.8	36200
159	United States	7.3	12.4	17.90	15.8	49400

	inflation	life_expec	total_fer	gdpp	class
7	1.160	82.0	1.93	51900	4
8	0.873	80.5	1.44	46900	4
15	1.880	80.0	1.86	44400	4
23	16.700	77.1	1.84	35300	4
29	2.870	81.3	1.63	47400	4
42	2.010	79.9	1.42	30800	4
44	3.220	79.5	1.87	58000	4
53	0.351	80.0	1.87	46200	4
54	1.050	81.4	2.03	40600	4
58	0.758	80.1	1.39	41800	4
60	0.673	80.4	1.48	26900	4
68	5.470	82.0	2.20	41900	4
73	-3.220	80.4	2.05	48700	4
74	1.770	81.4	3.03	30600	4
75	0.319	81.7	1.46	35800	4
77	-1.900	82.8	1.39	44500	4
82	11.200	78.2	2.21	38500	4
110	0.848	80.7	1.79	50300	4
111	3.730	80.9	2.17	33700	4

```
114       5.950        81.0        1.95  87800       4
122       0.643        79.8        1.39  22500       4
123       6.980        79.5        2.07  70300       4
135      -0.987        79.5        1.57  23400       4
138       3.160        80.1        1.23  22100       4
139       0.160        81.9        1.37  30700       4
144       0.991        81.5        1.98  52100       4
145       0.317        82.2        1.52  74600       4
157      12.500        76.5        1.87  35000       4
158       1.570        80.3        1.92  38900       4
159       1.220        78.7        1.93  48400       4
```

[30]:
```
df[df['class']==2]
```

[30]:
```
        country  child_mort  exports  health  imports  income  inflation  \
91   Luxembourg         2.8    175.0    7.77    142.0   91700      3.620
98        Malta         6.8    153.0    8.65    154.0   28300      3.830
133   Singapore         2.8    200.0    3.96    174.0   72100     -0.046

     life_expec  total_fer    gdpp  class
91         81.3       1.63  105000      2
98         80.3       1.36   21100      2
133        82.7       1.15   46600      2
```

[31]:
```
df[df['class']==0]
```

[31]:
```
       country  child_mort  exports  health  imports  income  inflation  \
103   Mongolia        26.1     46.7    5.44     56.7    7710       39.2
113    Nigeria       130.0     25.3    5.07     17.4    5150      104.0
163  Venezuela        17.1     28.5    4.91     17.6   16500       45.9

     life_expec  total_fer   gdpp  class
103        66.2       2.64   2650      0
113        60.5       5.84   2330      0
163        75.4       2.47  13500      0
```

[32]:
```
df[df['class']==3]
```

[32]:
```
                 country  child_mort  exports  health  imports  income  \
1                Albania        16.6     28.0    6.55     48.6    9930
2                Algeria        27.3     38.4    4.17     31.4   12900
4    Antigua and Barbuda        10.3     45.5    6.03     58.9   19100
5              Argentina        14.5     18.9    8.10     16.0   18700
6                Armenia        18.1     20.8    4.40     45.3    6700
..                   …          …        …       …        …       …
156              Ukraine        11.7     47.1    7.72     51.1    7820
160              Uruguay        10.6     26.3    8.35     25.4   17100
```

```
161              Uzbekistan        36.3      31.7     5.81      28.5     4240
162                 Vanuatu        29.2      46.6     5.25      52.7     2950
164                 Vietnam        23.3      72.0     6.84      80.2     4490

     inflation  life_expec  total_fer   gdpp  class
1         4.49        76.3       1.65   4090      3
2        16.10        76.5       2.89   4460      3
4         1.44        76.8       2.13  12200      3
5        20.90        75.8       2.37  10300      3
6         7.77        73.3       1.69   3220      3
..         ...         ...        ...    ...    ...
156      13.40        70.4       1.44   2970      3
160       4.91        76.4       2.08  11900      3
161      16.50        68.8       2.34   1380      3
162       2.62        63.0       3.50   2970      3
164      12.10        73.1       1.95   1310      3

[83 rows x 11 columns]
```

[33]:
```python
df[df['class']==1]
```

[33]:
```
                      country  child_mort  exports  health  imports  income  \
0                 Afghanistan        90.2   10.000    7.58  44.9000    1610
3                      Angola       119.0   62.300    2.85  42.9000    5900
17                      Benin       111.0   23.800    4.10  37.2000    1820
21                   Botswana        52.5   43.600    8.30  51.3000   13300
25               Burkina Faso       116.0   19.200    6.74  29.6000    1430
26                    Burundi        93.6    8.920   11.60  39.2000     764
28                   Cameroon       108.0   22.200    5.13  27.0000    2660
31   Central African Republic       149.0   11.800    3.98  26.5000     888
32                       Chad       150.0   36.800    4.53  43.5000    1930
36                    Comoros        88.2   16.500    4.51  51.7000    1410
37           Congo, Dem. Rep.       116.0   41.100    7.91  49.6000     609
38                Congo, Rep.        63.9   85.100    2.46  54.7000    5190
40              Cote d'Ivoire       111.0   50.600    5.30  43.3000    2690
49          Equatorial Guinea       111.0   85.800    4.48  58.9000   33700
50                    Eritrea        55.2    4.790    2.66  23.3000    1420
55                      Gabon        63.7   57.700    3.50  18.9000   15400
56                     Gambia        80.3   23.800    5.69  42.7000    1660
59                      Ghana        74.7   29.500    5.22  45.9000    3060
63                     Guinea       109.0   30.300    4.93  43.2000    1190
64              Guinea-Bissau       114.0   14.900    8.50  35.2000    1390
66                      Haiti       208.0   15.300    6.91  64.7000    1500
72                       Iraq        36.9   39.400    8.41  34.1000   12700
80                      Kenya        62.2   20.700    4.75  33.6000    2480
81                   Kiribati        62.7   13.300   11.30  79.9000    1730
84                        Lao        78.9   35.400    4.47  49.3000    3980
```

87	Lesotho	99.7	39.400	11.10	101.0000	2380
88	Liberia	89.3	19.100	11.80	92.6000	700
93	Madagascar	62.2	25.000	3.77	43.0000	1390
94	Malawi	90.5	22.800	6.59	34.9000	1030
97	Mali	137.0	22.800	4.98	35.1000	1870
99	Mauritania	97.4	50.700	4.41	61.2000	3320
106	Mozambique	101.0	31.500	5.21	46.2000	918
107	Myanmar	64.4	0.109	1.97	0.0659	3720
108	Namibia	56.0	47.800	6.78	60.7000	8460
112	Niger	123.0	22.200	5.16	49.1000	814
116	Pakistan	92.1	13.500	2.20	19.4000	4280
126	Rwanda	63.6	12.000	10.50	30.0000	1350
129	Senegal	66.8	24.900	5.66	40.3000	2180
132	Sierra Leone	160.0	16.800	13.10	34.5000	1220
136	Solomon Islands	28.1	49.300	8.55	81.2000	1780
137	South Africa	53.7	28.600	8.94	27.4000	12000
142	Sudan	76.7	19.700	6.32	17.2000	3370
147	Tanzania	71.9	18.700	6.01	29.1000	2090
149	Timor-Leste	62.6	2.200	9.12	27.8000	1850
150	Togo	90.3	40.200	7.65	57.3000	1210
155	Uganda	81.0	17.100	9.01	28.6000	1540
165	Yemen	56.3	30.000	5.18	34.4000	4480
166	Zambia	83.1	37.000	5.89	30.9000	3280

	inflation	life_expec	total_fer	gdpp	class
0	9.440	56.2	5.82	553	1
3	22.400	60.1	6.16	3530	1
17	0.885	61.8	5.36	758	1
21	8.920	57.1	2.88	6350	1
25	6.810	57.9	5.87	575	1
26	12.300	57.7	6.26	231	1
28	1.910	57.3	5.11	1310	1
31	2.010	47.5	5.21	446	1
32	6.390	56.5	6.59	897	1
36	3.870	65.9	4.75	769	1
37	20.800	57.5	6.54	334	1
38	20.700	60.4	4.95	2740	1
40	5.390	56.3	5.27	1220	1
49	24.900	60.9	5.21	17100	1
50	11.600	61.7	4.61	482	1
55	16.600	62.9	4.08	8750	1
56	4.300	65.5	5.71	562	1
59	16.600	62.2	4.27	1310	1
63	16.100	58.0	5.34	648	1
64	2.970	55.6	5.05	547	1
66	5.450	32.1	3.33	662	1
72	16.600	67.2	4.56	4500	1

```
80        2.090          62.8        4.37    967        1
81        1.520          60.7        3.84   1490        1
84        9.200          63.8        3.15   1140        1
87        4.150          46.5        3.30   1170        1
88        5.470          60.8        5.02    327        1
93        8.790          60.8        4.60    413        1
94       12.100          53.1        5.31    459        1
97        4.370          59.5        6.55    708        1
99       18.900          68.2        4.98   1200        1
106       7.640          54.5        5.56    419        1
107       7.040          66.8        2.41    988        1
108       3.560          58.6        3.60   5190        1
112       2.550          58.8        7.49    348        1
116      10.900          65.3        3.85   1040        1
126       2.610          64.6        4.51    563        1
129       1.850          64.0        5.06   1000        1
132      17.200          55.0        5.20    399        1
136       6.810          61.7        4.24   1290        1
137       6.350          54.3        2.59   7280        1
142      19.600          66.3        4.88   1480        1
147       9.250          59.3        5.43    702        1
149      26.500          71.1        6.23   3600        1
150       1.180          58.7        4.87    488        1
155      10.600          56.8        6.15    595        1
165      23.600          67.5        4.67   1310        1
166      14.000          52.0        5.40   1460        1
```

Let us provide the names to the labels.

0 - First Priority Nations 1 - Second Priority Nations 2 - Very Well Developed 3 - Developing Nations 4 - Well Developed

[34]:
```
df['class'] = df['class'].astype(str)

#make a new column called class names
df['class_name'] = df['class']

#add a new column with class descriptions called class_name
df['class_name'] = df['class_name'].replace({'0':'First Priority Nations','1':
↪'Second Priority Nations','2':'Very Well Developed','3':'Developing␣
↪Nations','4':'Well Developed'})
```

[35]:
```
df.head()
```

[35]:
```
         country  child_mort  exports  health  imports  income  \
0    Afghanistan        90.2     10.0    7.58     44.9    1610
1        Albania        16.6     28.0    6.55     48.6    9930
2        Algeria        27.3     38.4    4.17     31.4   12900
```

```
3                Angola       119.0      62.3     2.85      42.9     5900
4   Antigua and Barbuda        10.3      45.5     6.03      58.9    19100

   inflation  life_expec  total_fer   gdpp  class              class_name
0       9.44        56.2       5.82    553      1  Second Priority Nations
1       4.49        76.3       1.65   4090      3       Developing Nations
2      16.10        76.5       2.89   4460      3       Developing Nations
3      22.40        60.1       6.16   3530      1  Second Priority Nations
4       1.44        76.8       2.13  12200      3       Developing Nations
```

[36]:
```
#adding the className column to the scaled data

df_scaled['class_name'] = df['class_name']
```

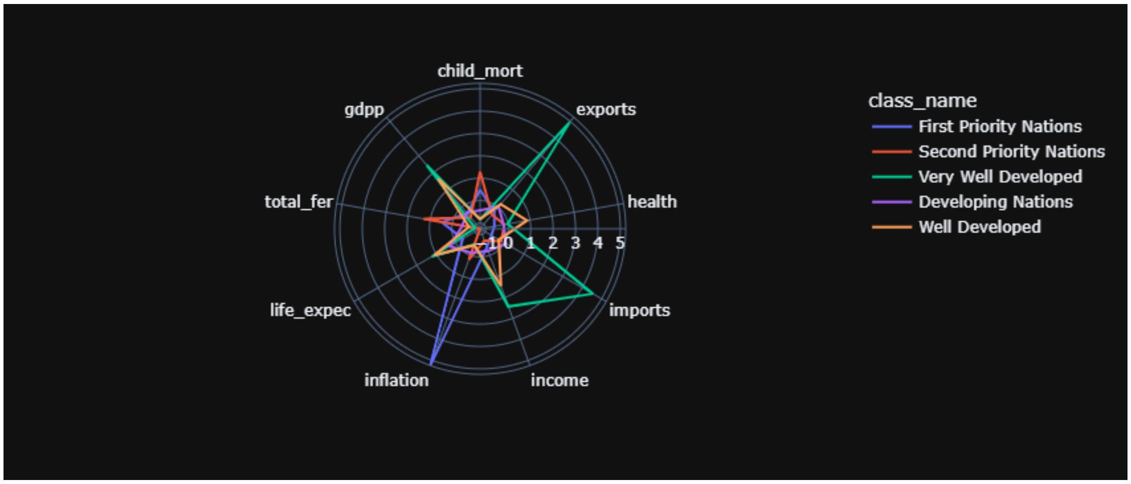
[37]:
```
#Again create a polar chart to visulize the countries acoording to class names
#grouping by class and getting mean values
polar_new = df_scaled.groupby('class').mean().reset_index()

#converting the dataframe
polar_new = pd.melt(polar_new, id_vars=['class'])

#creating a new column class_name in polar_new
polar_new['class_name'] = polar_new['class']

#changing the data type of class_name to object
polar_new['class_name'] = polar_new['class_name'].astype(str)

#replacing the categorical values to class names
polar_new['class_name'] = polar_new['class_name'].replace({'0':'First Priority␣
↪Nations','1':'Second Priority Nations','2':'Very Well Developed','3':
↪'Developing Nations','4':'Well Developed'})

polar_new
```

[37]:
```
    class    variable     value               class_name
0       0  child_mort  0.484065    First Priority Nations
1       1  child_mort  1.292620   Second Priority Nations
2       2  child_mort -0.849003       Very Well Developed
3       3  child_mort -0.434852        Developing Nations
4       4  child_mort -0.828609            Well Developed
5       0     exports -0.278413    First Priority Nations
6       1     exports -0.441377   Second Priority Nations
7       2     exports  4.935673       Very Well Developed
8       3     exports  0.024526        Developing Nations
9       4     exports  0.172621            Well Developed
10      0      health -0.611878    First Priority Nations
11      1      health -0.163124   Second Priority Nations
```

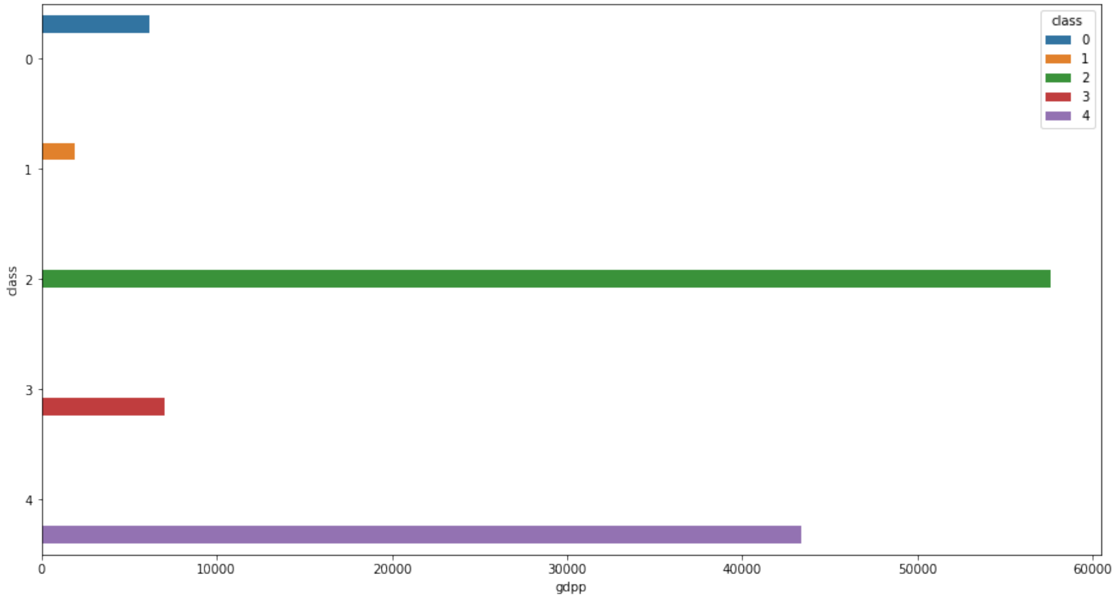
```
12       2       health -0.008163      Very Well Developed
13       3       health -0.193803       Developing Nations
14       4       health  0.859190           Well Developed
15       0      imports -0.676287   First Priority Nations
16       1      imports -0.170610  Second Priority Nations
17       2      imports  4.548058      Very Well Developed
18       3      imports  0.065845       Developing Nations
19       4      imports -0.296373           Well Developed
20       0       income -0.382826   First Priority Nations
21       1       income -0.690788  Second Priority Nations
22       2       income  2.439542      Very Well Developed
23       3       income -0.203380       Developing Nations
24       4       income  1.462275           Well Developed
25       0    inflation  5.242572   First Priority Nations
26       1    inflation  0.200143  Second Priority Nations
27       2    inflation -0.504206      Very Well Developed
28       3    inflation -0.114173       Developing Nations
29       4    inflation -0.478189           Well Developed
30       0   life_expec -0.359671   First Priority Nations
31       1   life_expec -1.261473  Second Priority Nations
32       2   life_expec  1.226824      Very Well Developed
33       3   life_expec  0.297828       Developing Nations
34       4   life_expec  1.107649           Well Developed
35       0    total_fer  0.465138   First Priority Nations
36       1    total_fer  1.306997  Second Priority Nations
37       2    total_fer -1.038863      Very Well Developed
38       3    total_fer -0.459087       Developing Nations
39       4    total_fer -0.763681           Well Developed
40       0         gdpp -0.372346   First Priority Nations
41       1         gdpp -0.606493  Second Priority Nations
42       2         gdpp  2.440797      Very Well Developed
43       3         gdpp -0.324708       Developing Nations
44       4         gdpp  1.661902           Well Developed
```

[47]:
```
%%capture
#plot the polar chart
fig = px.line_polar(polar_new, r="value", theta="variable", color="class_name", line_close=True, template="plotly_dark")
```

[48]:
```
fig.show()
```



[39]:
```
fig, ax = plt.subplots(figsize=(15,8))
df_bar = df.groupby('class').mean().reset_index()
fig = sns.barplot(x='gdpp', y='class',data=df_bar, hue='class')
```



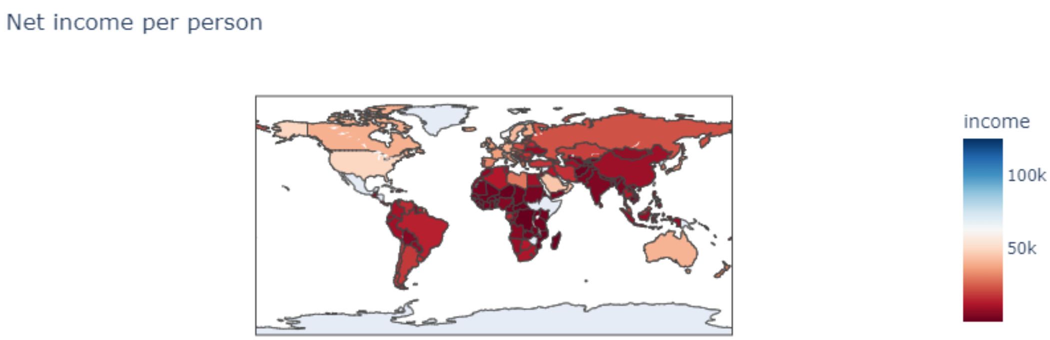
[ ]:
```
```

2 Socio Economic Factors

Some of the most important socio economic factors are child mortality rate, net income per person, health spending, import, export, GDP, etc. We will viusalize these factors for different countires below and try to compare with the clusters we have established.

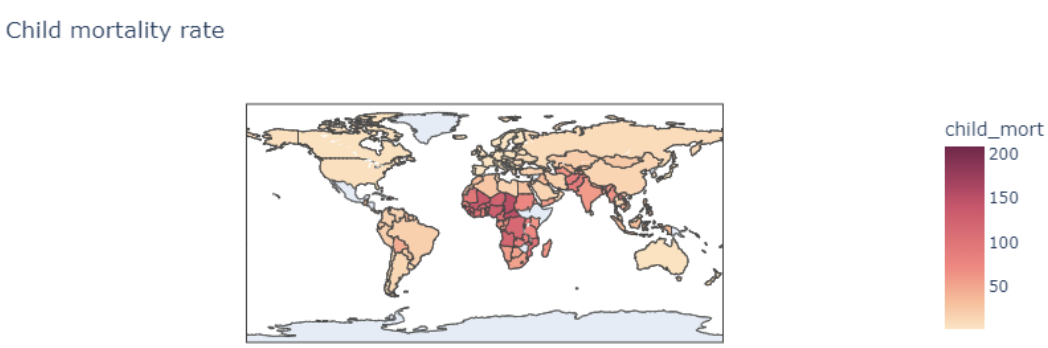
2.0.1 Net income per person

[40]:
```
px.choropleth(df, locationmode='country names',locations='country',␣
↪color='income',
                           color_continuous_scale="rdbu", title="Net income per␣
↪person"
                          )
```



2.1 Child Mortality rate

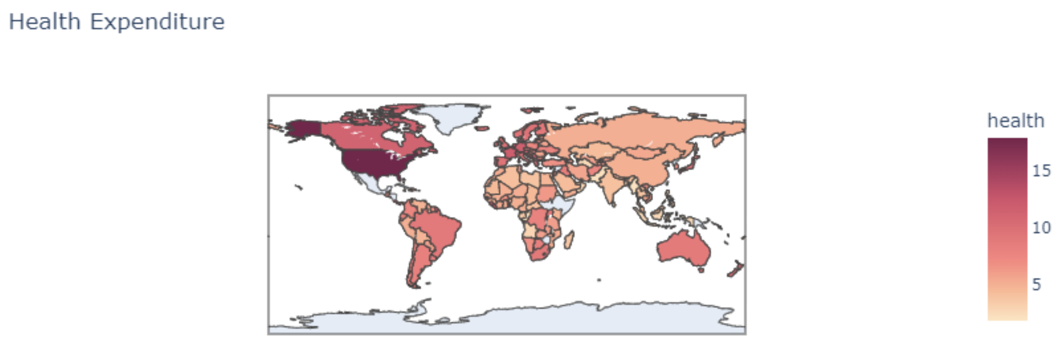
[41]:
```
px.choropleth(df, locationmode='country names',locations='country',␣
↪color='child_mort',
                           color_continuous_scale="burgyl", title="Child␣
↪mortality rate"
                          )
```



We can observe high mortality rate in African continent and then in south asian countries as well. How does it relate to health spending, we will visualize next.

2.1.1 Health Spending per Country

[42]:
```
px.choropleth(df, locationmode='country names',locations='country',␣
↪color='health',
                            color_continuous_scale="burgyl", title="Health␣
↪Expenditure"
                           )
```



Countires with highest health expenditure are from north america as well as western Europe. High levels of GDP per capita leads to higher budgets for health and in turn higher life expectancy.

2.2 Conclusion

1. DBSCAN takes care of the outliers, however it does not provide the best clusters of the countries as per the expectation.
2. DBSCAN also has low silhouette score compared to K-means
3. K means proved to be a better method to cluster the countries requiring the aid. Silhouette score was found to be greater than DBSCAN method and elbow method indicated a cluster of five.
4. The number of countries are divided into five categories with two of them requiring atmost attention in terms of financial aid.
5. There are 3 countries that showed very high inflation rate, lowest GDP and have been categorized as First Priority nations.

To Summarize following are the countries that need the attention and financial aid.

```
[43]: df[(df['class_name'] == 'First Priority Nations') | (df['class_name'] ==␣
      ↪'Second Priority Nations')].country
```

```
[43]: 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
      103                    Mongolia
      106                  Mozambique
      107                     Myanmar
      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
163                 Venezuela
165                     Yemen
166                    Zambia
Name: country, dtype: object
```

[ ]:

```

```