Yanis sindt-baret, Quentin Ohayon, Mayoran Raveendran

IA PROJECT

Attribute and object selection

Through this study, we chose to select a subset on a big databank on http://databank.worldbank.org. With this dataset, we want to determine the level of development of differents country thanks to our AI. To do that we first choose different attributes that could bring some informations to our model.

We first load all packages we will need

```
import pandas as pd
import matplotlib.pyplot as plt
import importlib
import data_managment as dm
import numpy as np
import seaborn
from outliers import smirnov_grubbs as grubbs
```

We start with the import of our dataframe, and we adapt it for our study

```
originial_df = pd.read_csv("data.txt", sep="\t")
originial_df.drop(["Country Code", "Time Code","Time"], axis=1, inplace = True)
originial_df.set_index("Country Name", inplace = True)
originial_df[originial_df == ".."] = np.nan
df = originial_df.copy().astype(np.float64)
```

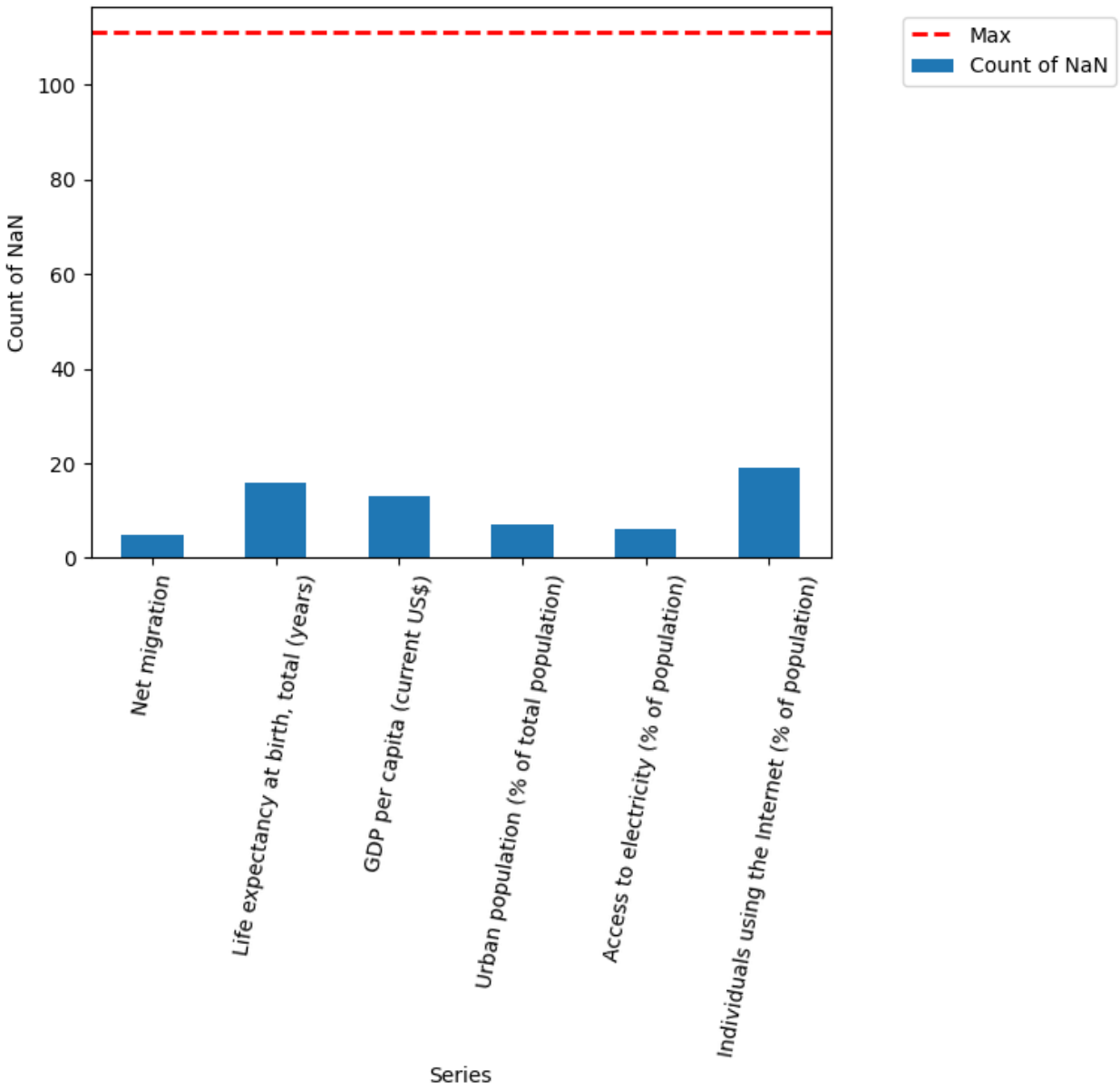
NaN purification

We can see there are many NaN : We can choose first to eliminate country with many NaN data (over 50% for example) that can bring the outlier. But that value can be choosing with the following graphic

```
df_nb_nan = dm.make_na_count(df,True); print(df_nb_nan)
df_nb_nan["Series"] = df_nb_nan.index
df_nb_nan.plot.bar(y="NaN_count", x = "Series",  label="Count of NaN", rot=80)
max_nan = df.shape[0]*0.5
plt.axhline(max_nan, color='r', linestyle='dashed', linewidth=2, label="Max")
plt.ylabel("Count of NaN")
plt.legend(loc='upper right', bbox_to_anchor=(1.4, 1))
```

```
                                                NaN_count
Net migration                                           5
Life expectancy at birth, total (years)                16
GDP per capita (current US$)                           13
Urban population (% of total population)                7
Access to electricity (% of population)                 6
Individuals using the Internet (% of population)       19
```

```
<matplotlib.legend.Legend at 0x132f2846e50>
```



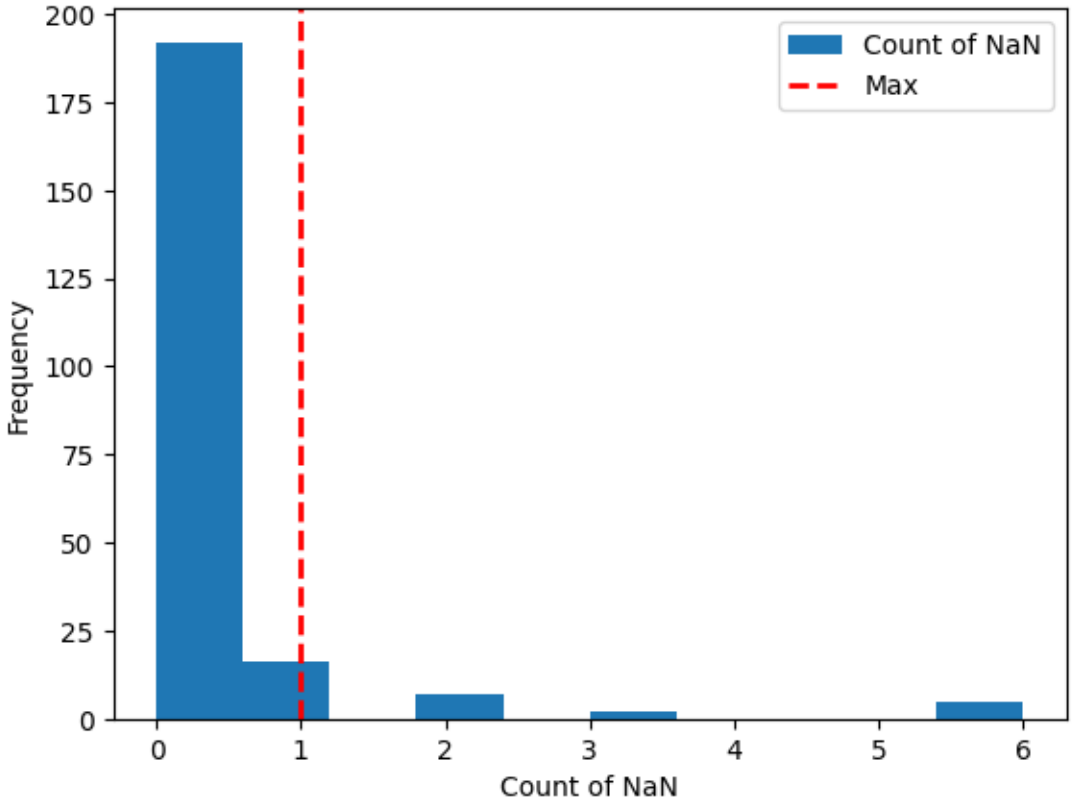
So we delete all series with more that 50% of data corresponding to NaN values, meaning that the attributes don't bring enough information to be interesting. Here, all attributes are kept.

```
df_without_na_series = dm.del_many_na_series(df, df_nb_nan, max_nan)
```

Now, we will check if some country still go NaN values ;

```
df_nb_nan_s = dm.make_na_count(df,False)
df_nb_nan_s["Series"] = df_nb_nan_s.index
df_nb_nan_s.plot.hist(y="NaN_count",  label="Count of NaN")
max_nan_s = 1
plt.axvline(max_nan_s, color='r', linestyle='dashed', linewidth=2, label="Max")
plt.xlabel("Count of NaN")
plt.legend(loc='upper right')
```

```
<matplotlib.legend.Legend at 0x132f4262310>
```



Few country has been removed because still presence of NaN values.

```
df_without_na_country = dm.del_many_na_country(df_without_na_series, df_nb_nan_s, max_nan_s)\
;df_without_na_country
```

```
                      Net migration  Life expectancy at birth, total (years)  \
Country Name
Afghanistan                -90238.0                                63.136000
Albania                    -10887.0                                78.860000
Algeria                    -36227.0                                75.732000
Angola                      91623.0                                61.092000
Antigua and Barbuda            16.0                                78.152000
...                             ...                                      ...
Virgin Islands (U.S.)         -526.0                                79.168293
West Bank and Gaza         -22774.0                                74.554000
Yemen, Rep.                -39688.0                                66.064000
Zambia                      17285.0                                61.794000
Zimbabwe                   -59918.0                                60.306000

                      GDP per capita (current US$)  \
Country Name
Afghanistan                            520.252064
Albania                               4124.055390
Algeria                               3967.199451
Angola                                1709.515534
Antigua and Barbuda                  15862.651663
...                                           ...
Virgin Islands (U.S.)                35324.974887
West Bank and Gaza                    3527.613824
Yemen, Rep.                           1069.817122
Zambia                                1249.923143
```

```
Zimbabwe                                1421.787789

                      Urban population (% of total population)  \
Country Name
Afghanistan                                                25.020
Albania                                                    58.421
Algeria                                                    71.459
Angola                                                     64.149
Antigua and Barbuda                                        24.846
...                                                           ...
Virgin Islands (U.S.)                                      95.480
West Bank and Gaza                                         75.628
Yemen, Rep.                                                35.394
Zambia                                                     42.438
Zimbabwe                                                   32.296

                      Access to electricity (% of population)  \
Country Name
Afghanistan                                           97.699997
Albania                                               99.889999
Algeria                                               99.350250
Angola                                                41.813129
Antigua and Barbuda                                  100.000000
...                                                         ...
Virgin Islands (U.S.)                                100.000000
West Bank and Gaza                                   100.000000
Yemen, Rep.                                           68.859505
Zambia                                                35.425453
Zimbabwe                                              42.561729

                      Individuals using the Internet (% of population)
Country Name
Afghanistan                                                  11.000000
Albania                                                      59.600000
Algeria                                                      42.945527
Angola                                                       23.200000
Antigua and Barbuda                                          73.000000
...                                                                ...
Virgin Islands (U.S.)                                        59.608316
West Bank and Gaza                                           59.900000
Yemen, Rep.                                                  24.579208
Zambia                                                       10.300000
Zimbabwe                                                     23.119989

[192 rows x 6 columns]
```

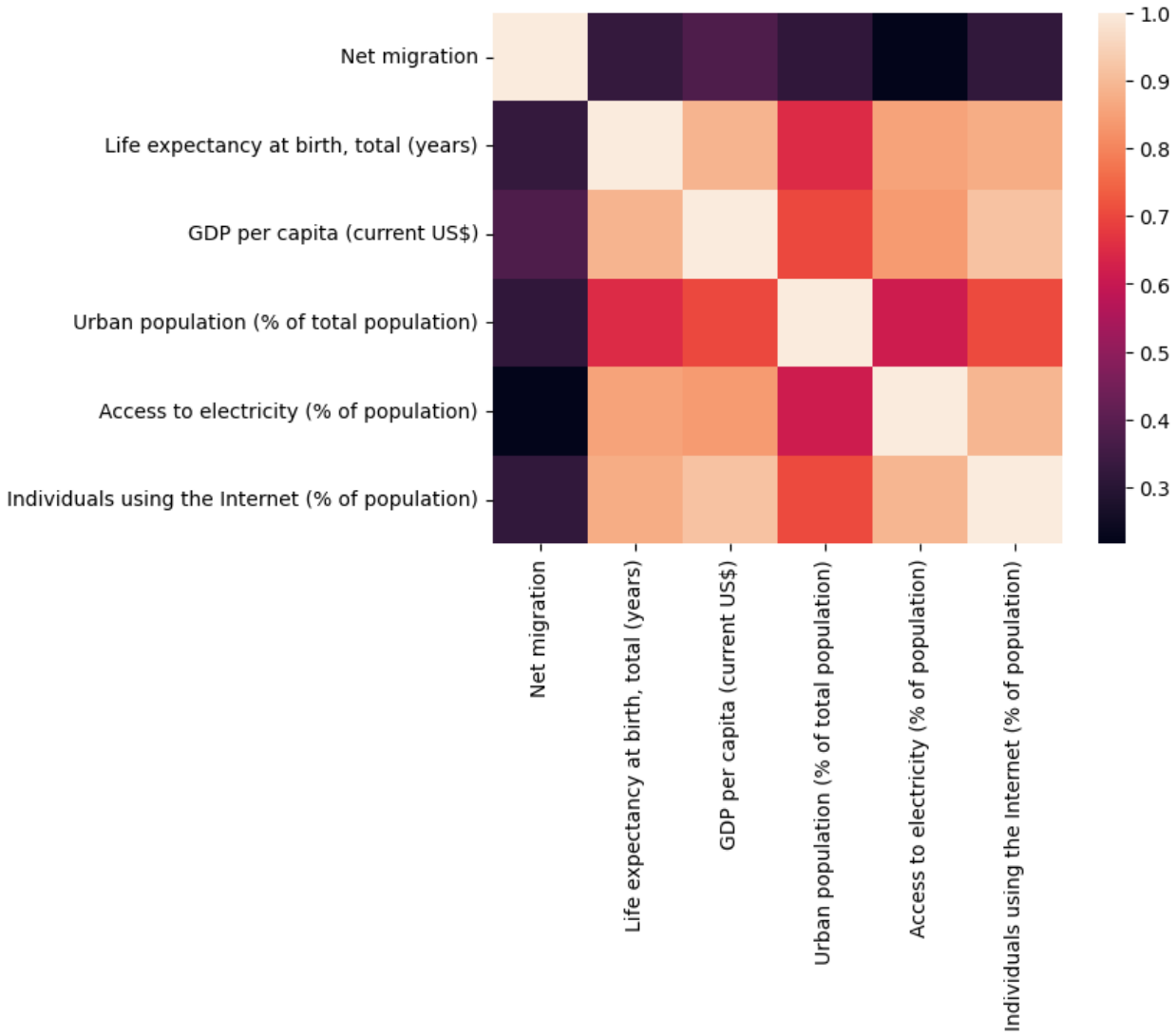
But we can also replace NaN value by using a clustering estimation with K nearest neighbours methods. But we decided don't do it, because it could change the behaviour to our data, and after few try, we did not had convincing result.

Correleted data

We will now check about coreleted data with spearman method.

```
df_corr = df_without_na_country.corr("spearman", numeric_only = True)
seaborn.heatmap(df_corr)
```

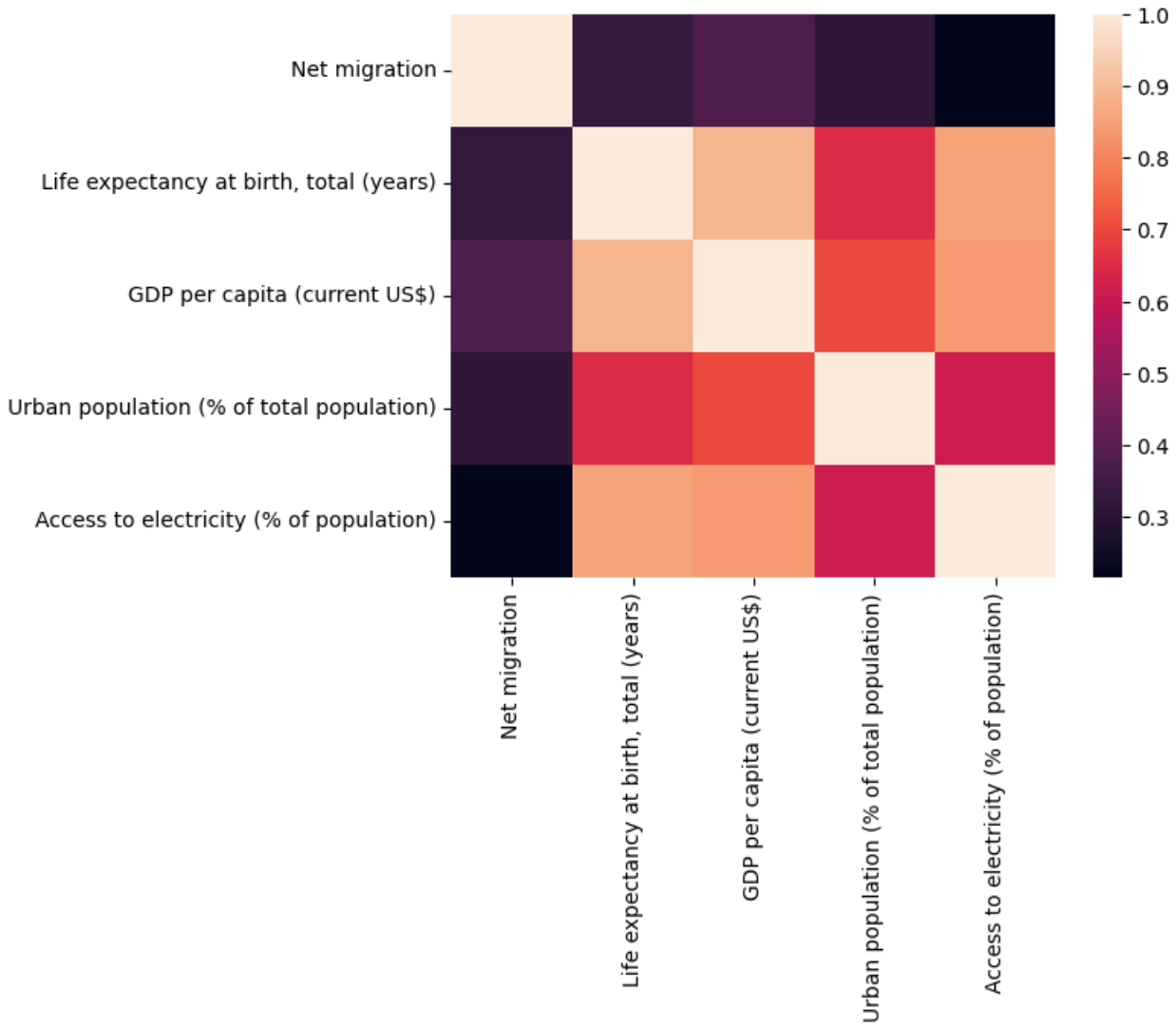
```
<AxesSubplot: >
```



We this heatmap about contengensis table establish the different possible relation, that we will delete one of the variable when there is 90% absolute correlation effect. That will be done for both dataframe we have.

```
df_without_corr_series = dm.del_correled_series(df_without_na_country, df_corr,0.90)
corr_clean = df_without_corr_series.corr("spearman")
seaborn.heatmap(corr_clean)
```

```
<AxesSubplot: >
```



We see that acces to individuals using internet has been deleted because of its correlation with acces to eletricity. That seems logical, because internet use depends of the acces of eletricity, and most of the time, when eletricity is accessible, internet is also.

Normalization of data with z-score

At this step, all data that we considered as useless (with NaN values, or correlated data) have deleted. Because of the variety of our variable units, but also order of magnitude. We should normalize ours. After some research and also internal tries, we saw that normalized data brings much more information for clustering and classification.

To make easier the use of our data, we create csv files countening it.

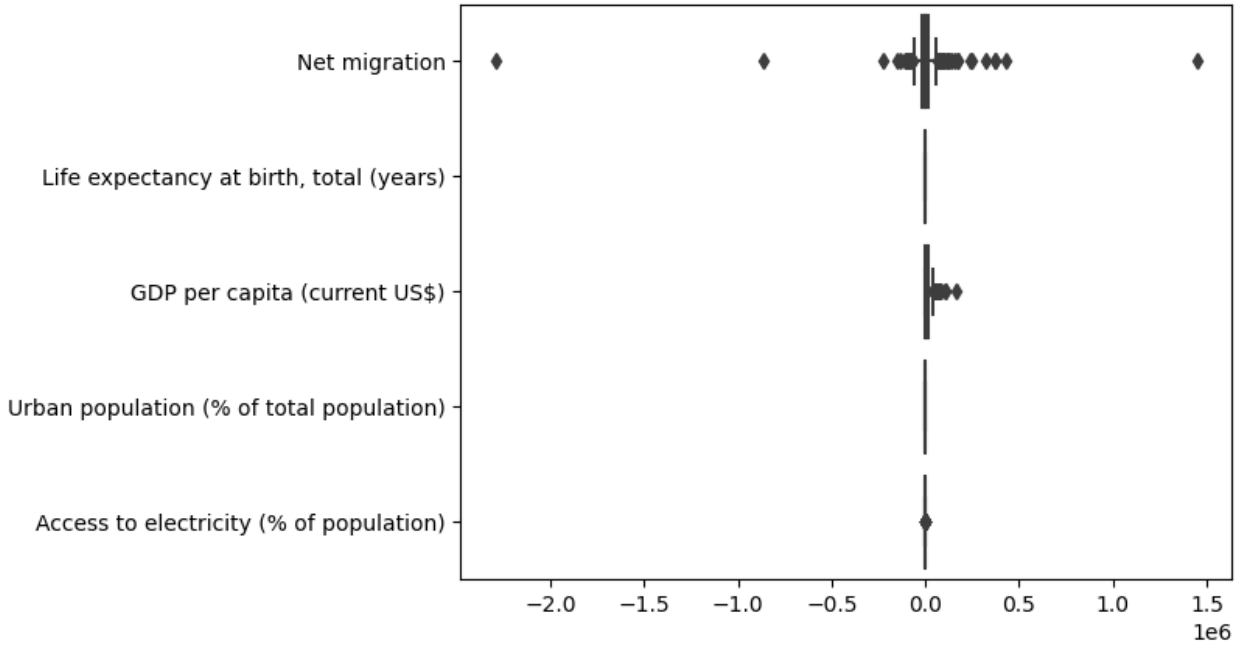
```
n_df = dm.normalize_df(df_without_corr_series)
dm.export_clean_data(df_without_corr_series)
dm.export_clean_data(n_df,"normalized_clean_dataframe.csv")
```

Check of Outliers

To finish the evaluation of our dataset, outliers can be searched and determined first graphically, and then thanks to statistical test as the grubbs test

```
seaborn.boxplot(data=df_without_corr_series, orient = "h")
```

```
<AxesSubplot: >
```

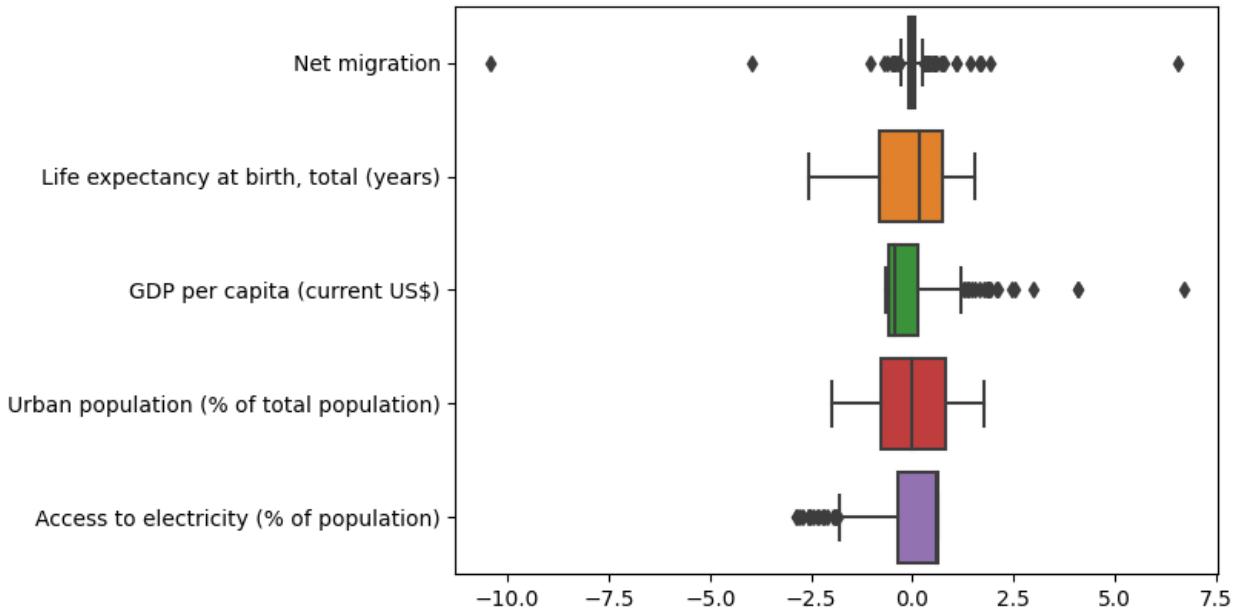


We can see on the plot that there are some outliers on the net migration, and also for GDP per capita. We can check thanks to the grubbs text, how much there are, that is significant.

```
seaborn.boxplot(data=n_df, orient = "h")
```

```
outliers_data = {}
for serie in n_df:
    val = np.array(n_df[serie])
    outl = grubbs.max_test_indices(val, alpha=0.05) + grubbs.max_test_indices(-val, alpha=0.05)
    print(f"We have found {len(outl)} outliers in the series : {serie}")
    outliers_data[serie] = outl
```

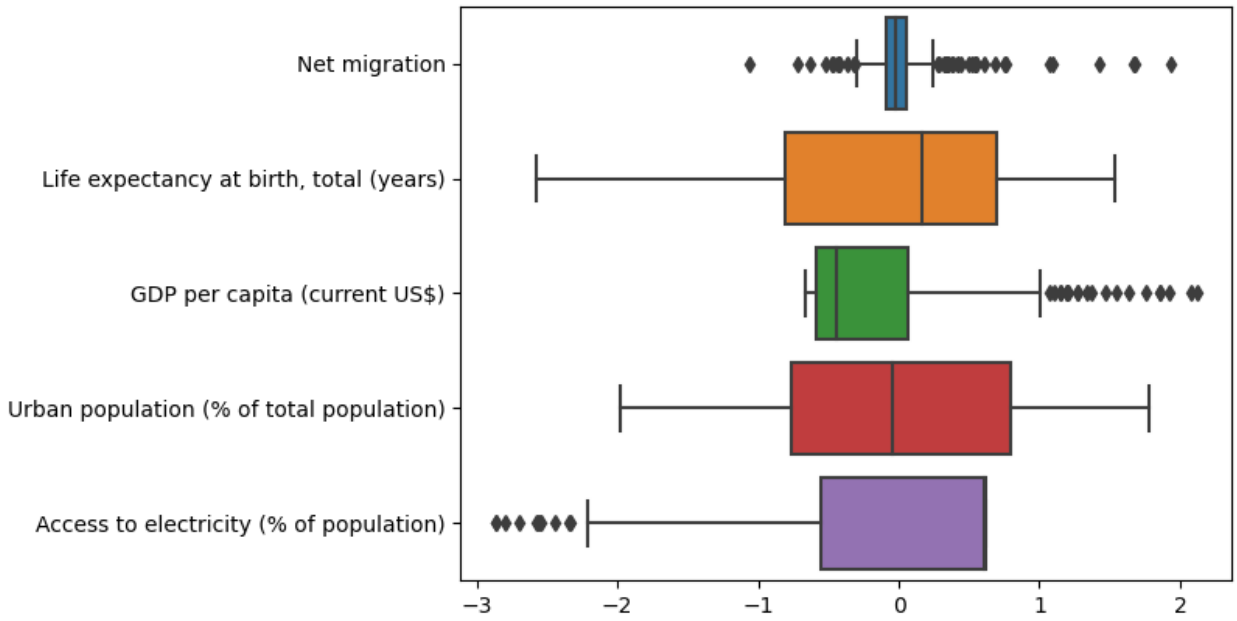
```
We have found 3 outliers in the series : Net migration
We have found 0 outliers in the series : Life expectancy at birth, total (years)
We have found 6 outliers in the series : GDP per capita (current US$)
We have found 0 outliers in the series : Urban population (% of total population)
We have found 0 outliers in the series : Access to electricity (% of population)
```



So, the grubbs test give us 9 outliers on our subdataset. We chose to don't change the reality of our data and so, to not delete them. But we are making an other dataframe, where we will delete them. Then, both dataframes will be compared, to see if the impact of the outliers are too significant, and then choose or not to use the dataframe with, ou without these point for clusturing or classification.

To make the new dataframe without outliers we do :

```
n_df_wo_outliar = n_df.copy()
for series in outliers_data.values():
    for val in series:
        try:
            n_df_wo_outliar.drop(df_without_corr_series.iloc[val].name, axis = 0, inplace=True)
        except KeyError:
            pass

seaborn.boxplot(data=n_df_wo_outliar, orient = "h")
dm.export_clean_data(n_df_wo_outliar,"normalized_clean_dataframe_wo_outlier.csv")
```



So we have 3 differents subsets based a bigger dataframe (on a databank). We will start to study this dataset by making clustering of our data, and search if we can see differents level of development thanks to our attributes. We will then try to do a classification on this dataset (that will be on another notebook).

```
import pandas as pd
import numpy as np
from sklearn.cluster import DBSCAN
import seaborn as sns
import data_managment as dm
import importlib
import clustering_function as cf
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.impute import KNNImputer
from sklearn.tree import DecisionTreeClassifier
from sklearn.cluster import KMeans
from sklearn import metrics
import sklearn.model_selection
import scipy as sp
import scipy.cluster.hierarchy as sch
```

importing data

```
data = pd.read_csv("normalized_clean_dataframe.csv", index_col=0)
clean_data=pd.read_csv("normalized_clean_dataframe_wo_outlier.csv",index_col=0)
```

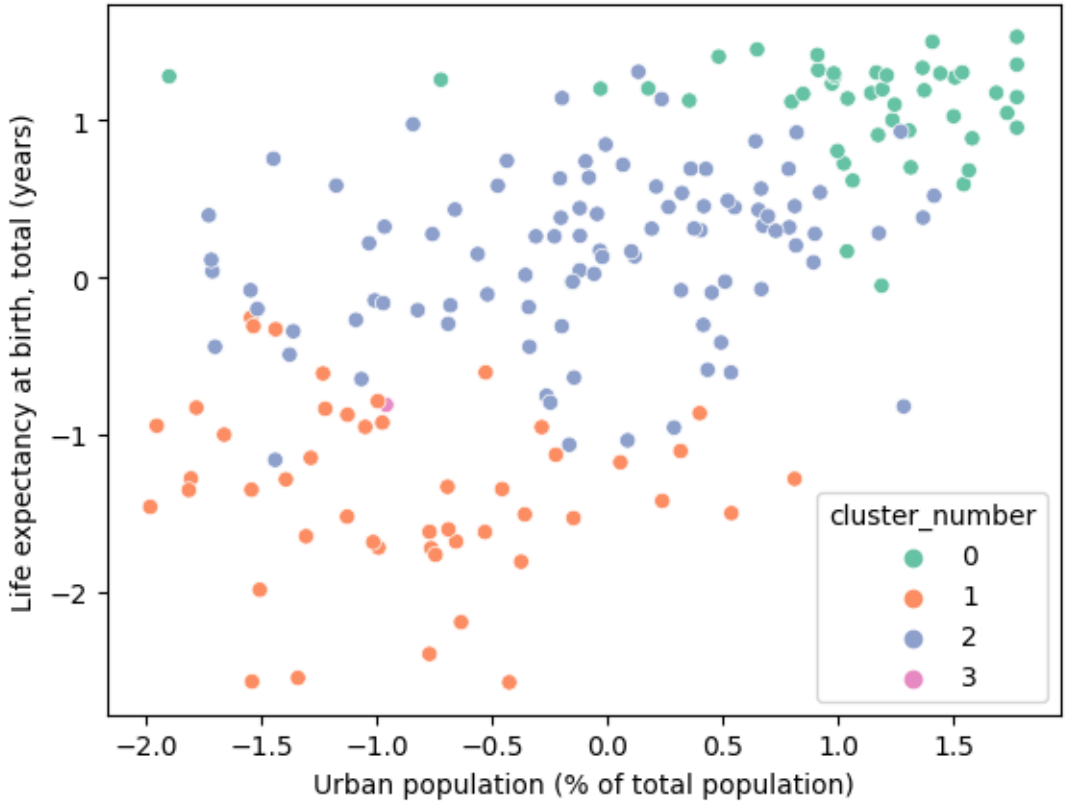
Data importation with and without outliers (data and clean_data) to study the effetcts of outliers on the quality of clustering.

Kmeans clustering

With outliers

```
df_and_cluster,clusters_km=cf.Kmeans_clustering(data,4)
sns.scatterplot(data=df_and_cluster, x="Urban population (% of total population)",
y="Life expectancy at birth, total (years)",hue="cluster_number",
palette="Set2", legend="full")
```

```
<AxesSubplot: xlabel='Urban population (% of total population)',
ylabel='Life expectancy at birth, total (years)'>
```



We can also show the differents country in each cluster with :

```
cf.print_all_content_cluster(df_and_cluster)
```

```
Cluster number 0
['Australia', 'Austria', 'Bahamas, The', 'Bahrain', 'Belgium', 'Bermuda', 'Canada',
'Chile', 'Curacao', 'Cyprus', 'Denmark', 'Faroe Islands', 'Finland', 'France', 'Germany',
'Greece', 'Greenland', 'Guam', 'Hong Kong SAR, China', 'Iceland', 'Ireland', 'Israel',
'Italy', 'Japan', 'Korea, Rep.', 'Kuwait', 'Liechtenstein', 'Luxembourg', 'Macao SAR,
China', 'Malta', 'Netherlands', 'New Zealand', 'Norway', 'Oman', 'Puerto Rico',
'Qatar', 'Saudi Arabia', 'Singapore', 'Spain', 'Sweden', 'Switzerland', 'United Arab Emirates',
'United Kingdom', 'United States', 'Uruguay', 'Virgin Islands (U.S.)']

----------------------------------------------------------
Cluster number 1
['Angola', 'Benin', 'Botswana', 'Burkina Faso', 'Burundi', 'Cambodia', 'Cameroon',
'Central African Republic', 'Chad', 'Comoros', 'Congo, Dem. Rep.', 'Congo, Rep.',
"Cote d'Ivoire", 'Djibouti', 'Equatorial Guinea', 'Eswatini', 'Ethiopia', 'Gambia,
The', 'Guinea', 'Guinea-Bissau', 'Haiti', 'Kenya', 'Lesotho', 'Liberia', 'Madagascar',
'Malawi', 'Mali', 'Mauritania', 'Mozambique', 'Myanmar', 'Namibia', 'Niger', 'Nigeria',
'Papua New Guinea', 'Rwanda', 'Senegal', 'Sierra Leone', 'Solomon Islands', 'Somalia',
'Sudan', 'Tanzania', 'Timor-Leste', 'Togo', 'Uganda', 'Vanuatu', 'Yemen, Rep.',
'Zambia', 'Zimbabwe']

----------------------------------------------------------
Cluster number 2
['Afghanistan', 'Albania', 'Algeria', 'Antigua and Barbuda', 'Argentina', 'Armenia',
'Aruba', 'Azerbaijan', 'Bangladesh', 'Barbados', 'Belarus', 'Belize', 'Bhutan',
'Bolivia', 'Bosnia and Herzegovina', 'Brazil', 'Brunei Darussalam', 'Bulgaria',
'Cabo Verde', 'China', 'Colombia', 'Costa Rica', 'Croatia', 'Cuba', 'Czechia',
```

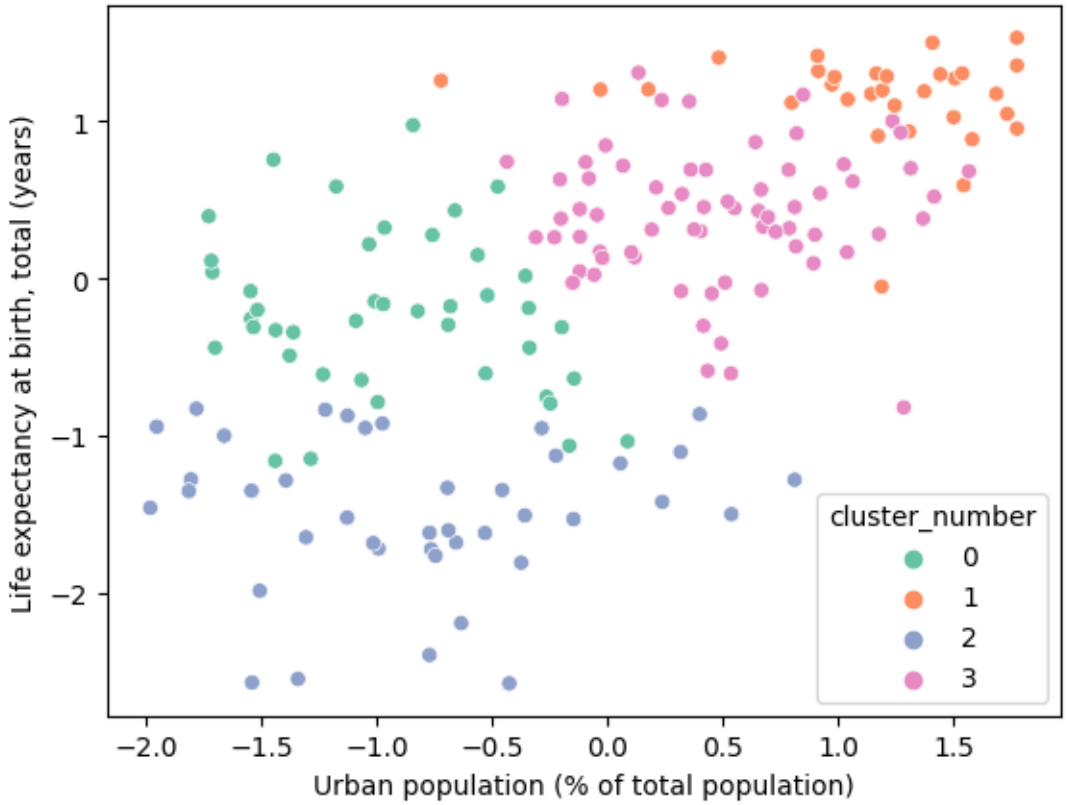
```
'Dominican Republic', 'Ecuador', 'Egypt, Arab Rep.', 'El Salvador', 'Estonia', 'Fiji',
'French Polynesia', 'Gabon', 'Georgia', 'Ghana', 'Grenada', 'Guatemala', 'Guyana',
'Honduras', 'Hungary', 'India', 'Indonesia', 'Iran, Islamic Rep.', 'Iraq', 'Jamaica',
'Jordan', 'Kazakhstan', 'Kiribati', 'Kyrgyz Republic', 'Lao PDR', 'Latvia', 'Lebanon',
'Lithuania', 'Malaysia', 'Maldives', 'Mauritius', 'Mexico', 'Micronesia, Fed. Sts.',
'Moldova', 'Mongolia', 'Montenegro', 'Morocco', 'Nepal', 'Nicaragua', 'North Macedonia',
'Panama', 'Paraguay', 'Peru', 'Philippines', 'Poland', 'Portugal', 'Romania', 'Russian Federation',
'Samoa', 'Sao Tome and Principe', 'Serbia', 'Seychelles', 'Slovak Republic', 'Slovenia',
'South Africa', 'Sri Lanka', 'St. Lucia', 'St. Vincent and the Grenadines', 'Suriname',
'Syrian Arab Republic', 'Tajikistan', 'Thailand', 'Tonga', 'Trinidad and Tobago',
'Tunisia', 'Turkiye', 'Turkmenistan', 'Tuvalu', 'Ukraine', 'Uzbekistan', 'Vietnam', 'West Bank and Gaza']
--------------------------------------------------------
Cluster number 3
['Pakistan']
--------------------------------------------------------
```

Since Kmeans clustering uses a predefined number of clusters, the presence of an outliers (Pakistan) in the dataset is occuping a cluster. Therefore, the clustering is less efficient and some countries that are not similar will be put into the same cluster.

Without outliers

```
df_and_cluster_wo_out,clusters_km_wo_out=cf.Kmeans_clustering(clean_data,4)
sns.scatterplot(data=df_and_cluster_wo_out, x="Urban population (% of total population)",
y="Life expectancy at birth, total (years)",hue="cluster_number",
palette="Set2", legend="full")
```

```
<AxesSubplot: xlabel='Urban population (% of total population)',
ylabel='Life expectancy at birth, total (years)'>
```




```
cf.print_all_content_cluster(df_and_cluster_wo_out)
```

```
Cluster number 0
```

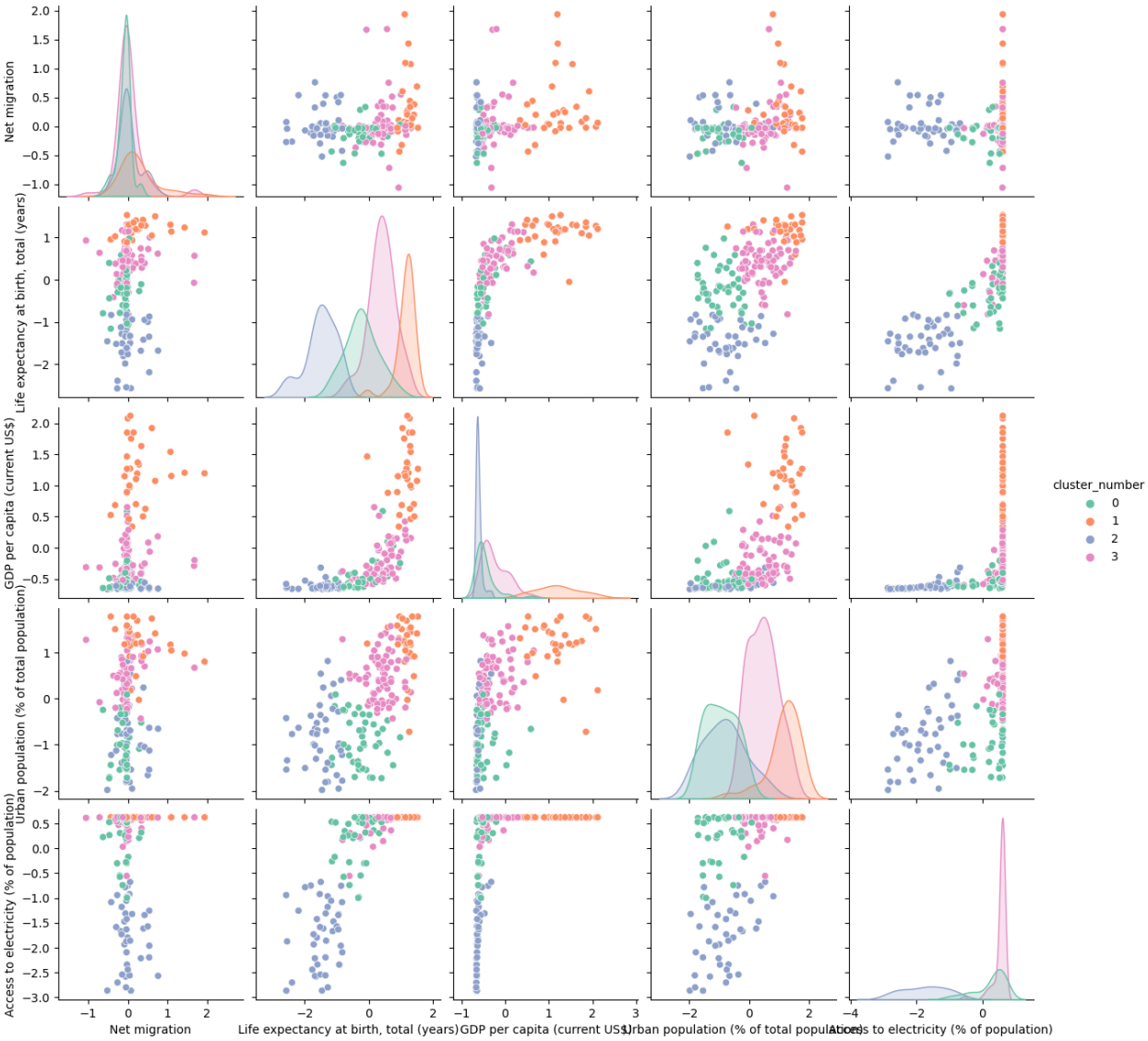
```
['Afghanistan', 'Antigua and Barbuda', 'Aruba', 'Bangladesh', 'Barbados', 'Belize',
'Bhutan', 'Bosnia and Herzegovina', 'Cambodia', 'Comoros', 'Egypt, Arab Rep.', 'Fiji',
'Ghana', 'Grenada', 'Guatemala', 'Guyana', 'India', 'Indonesia', 'Kiribati', 'Kyrgyz Republic',
'Lao PDR', 'Maldives', 'Mauritius', 'Micronesia, Fed. Sts.', 'Moldova', 'Nepal',
'Philippines', 'Samoa', 'Senegal', 'Solomon Islands', 'Sri Lanka', 'St. Lucia',
'Syrian Arab Republic', 'Tajikistan', 'Timor-Leste', 'Tonga', 'Turkmenistan', 'Tuvalu',
'Uzbekistan', 'Vanuatu', 'Vietnam', 'Yemen, Rep.']

----------------------------------------------------
Cluster number 1
['Australia', 'Austria', 'Bahrain', 'Belgium', 'Canada', 'Denmark', 'Faroe Islands',
'Finland', 'France', 'Germany', 'Greenland', 'Guam', 'Hong Kong SAR, China', 'Iceland',
'Ireland', 'Israel', 'Italy', 'Japan', 'Korea, Rep.', 'Kuwait', 'Malta', 'Netherlands',
'New Zealand', 'Puerto Rico', 'Qatar', 'Singapore', 'Spain', 'Sweden', 'United Arab Emirates',
'United Kingdom', 'Virgin Islands (U.S.)']

----------------------------------------------------
Cluster number 2
['Angola', 'Benin', 'Botswana', 'Burkina Faso', 'Burundi', 'Cameroon', 'Central African Republic',
'Chad', 'Congo, Dem. Rep.', 'Congo, Rep.', "Cote d'Ivoire", 'Djibouti', 'Equatorial Guinea',
'Eswatini', 'Ethiopia', 'Gambia, The', 'Guinea', 'Guinea-Bissau', 'Haiti', 'Kenya',
'Lesotho', 'Liberia', 'Madagascar', 'Malawi', 'Mali', 'Mauritania', 'Mozambique',
'Myanmar', 'Namibia', 'Niger', 'Nigeria', 'Papua New Guinea', 'Rwanda', 'Sierra Leone',
'Somalia', 'Sudan', 'Tanzania', 'Togo', 'Uganda', 'Zambia', 'Zimbabwe']

----------------------------------------------------
Cluster number 3
['Albania', 'Algeria', 'Argentina', 'Armenia', 'Azerbaijan', 'Bahamas, The', 'Belarus',
'Bolivia', 'Brazil', 'Brunei Darussalam', 'Bulgaria', 'Cabo Verde', 'Chile', 'China',
'Colombia', 'Costa Rica', 'Croatia', 'Cuba', 'Curacao', 'Cyprus', 'Czechia', 'Dominican Republic',
'Ecuador', 'El Salvador', 'Estonia', 'French Polynesia', 'Gabon', 'Georgia', 'Greece',
'Honduras', 'Hungary', 'Iran, Islamic Rep.', 'Iraq', 'Jamaica', 'Jordan', 'Kazakhstan',
'Latvia', 'Lebanon', 'Lithuania', 'Malaysia', 'Mexico', 'Mongolia', 'Montenegro',
'Morocco', 'Nicaragua', 'North Macedonia', 'Oman', 'Panama', 'Paraguay', 'Peru',
'Poland', 'Portugal', 'Romania', 'Russian Federation', 'Sao Tome and Principe',
'Saudi Arabia', 'Serbia', 'Seychelles', 'Slovak Republic', 'Slovenia', 'St. Vincent and the Grenadines',
'Suriname', 'Thailand', 'Trinidad and Tobago', 'Tunisia', 'Turkiye', 'Ukraine',
'Uruguay', 'West Bank and Gaza']

----------------------------------------------------
```

The distribution of countries in each cluster is more balanced. We can observe some similarities between each clusters. The countries in clusters 0 have a relatively low urban population and decent life expectancy, they corresponds to future emerging countries. There are many islands in this group. In cluster 1, countries have high life expectancy and urban population, they are developed countries. The countries in cluster 2 are the polar opposite of those in 1, they belongs to the under developed countries. In the cluster 3, the countries have intermediate coordinates, they are emerging countries. The Kmeans clustering without outliers allows to group the countries according to their level of development. That is why we will keep the dataset without outliers. However, we need to keep in mind that removing some points will still have an impact on the conclusions we may draw, as we lose some information.

The pairplot allows us to see the clusters projected on every combinaisons of 2 axes, to find the one which is the more readable.

```
sns.pairplot(df_and_cluster_wo_out, hue="cluster_number",palette="Set2")
```

```
<seaborn.axisgrid.PairGrid at 0x132f4fa7150>
```

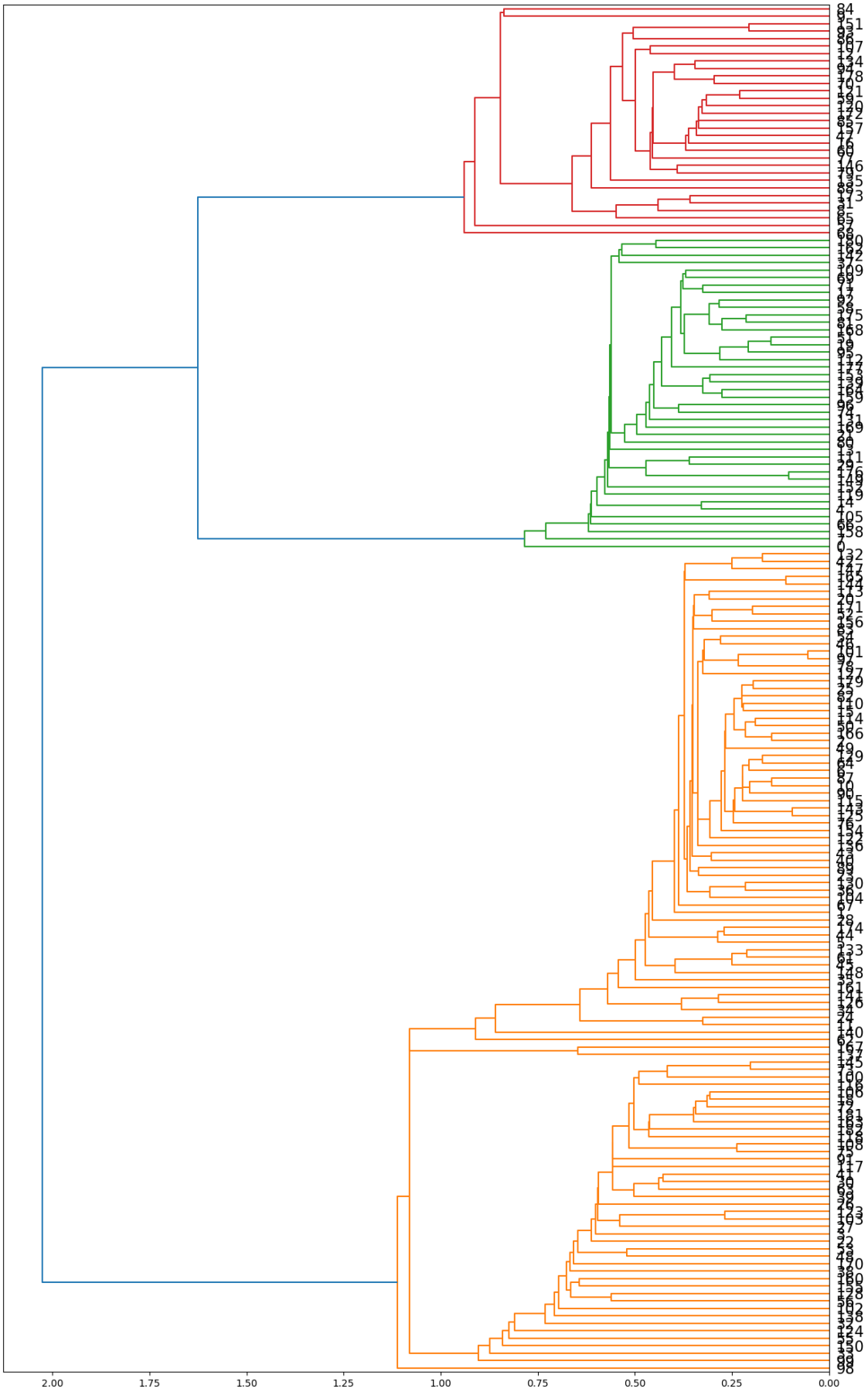


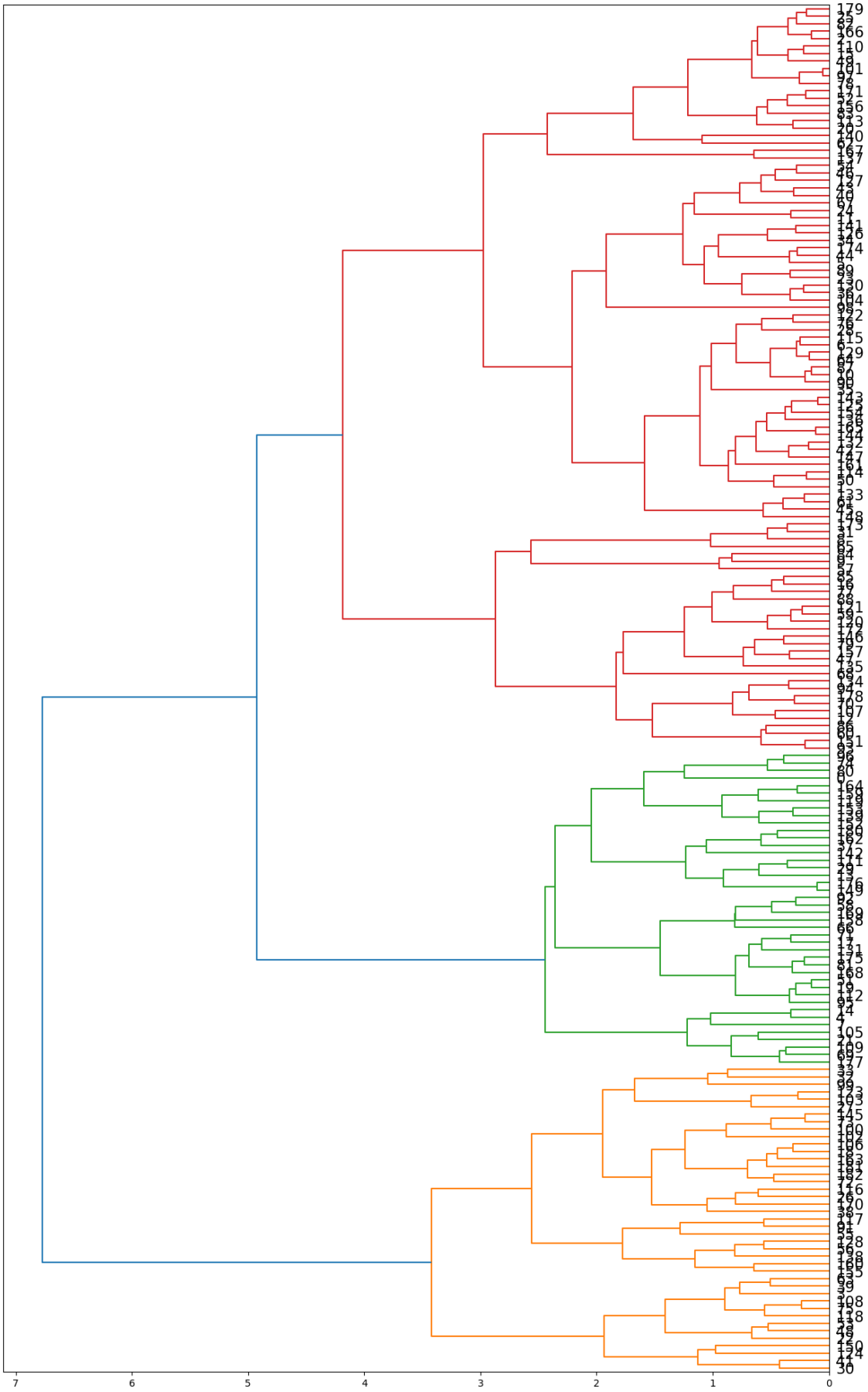
The projection on Urban population and Life expectancy seems to be the best to visualize the data.

Hierarchical clustering

We compare 2 methods of hierarchical clustering (single and complete) to find the more suitable in our case. Moreover, the results are also compared to the clusters obtained by Kmeans.

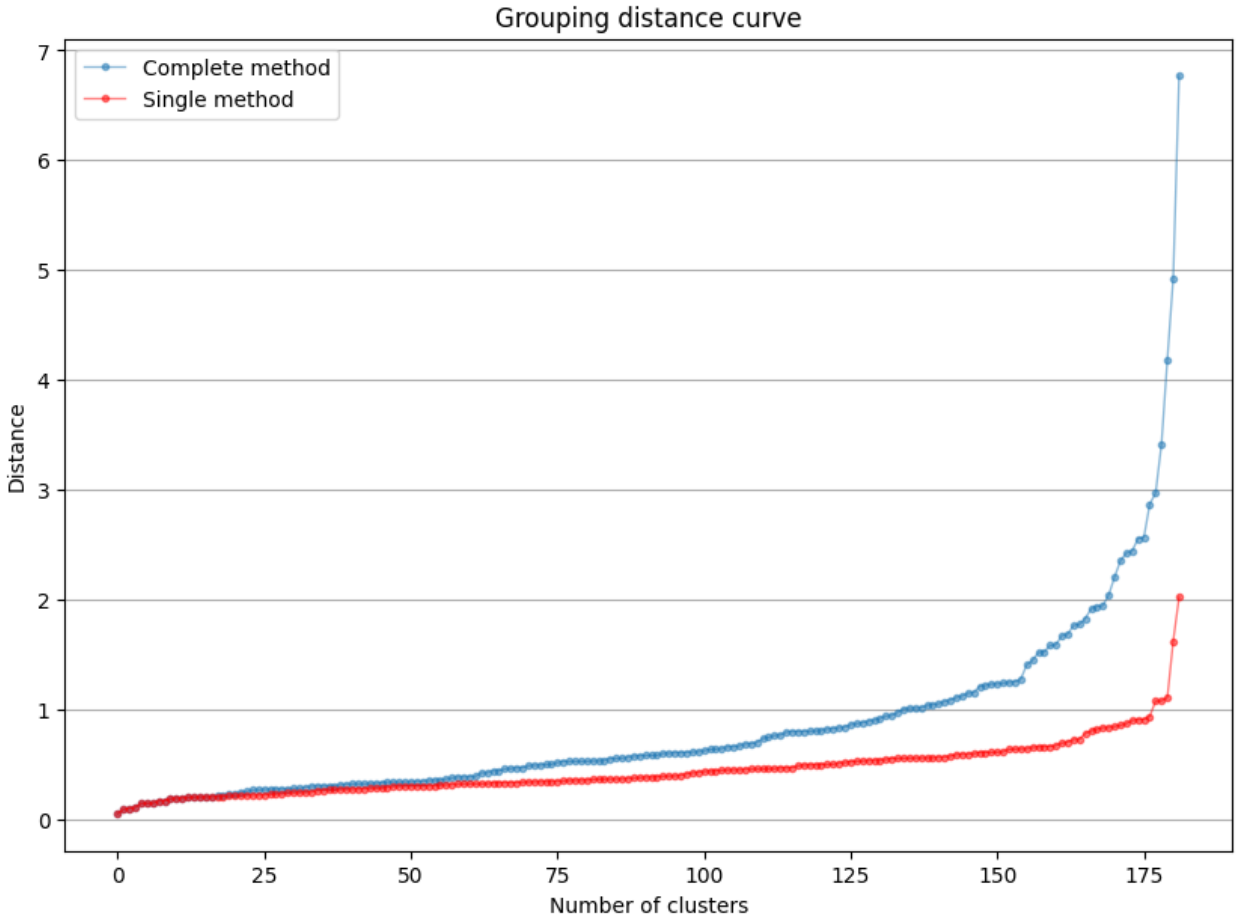
```
Z_single,clusters_hs= cf.hierarchical_clustering(clean_data,4,method='single')
Z_complete,clusters_hc=cf.hierarchical_clustering(clean_data,4)
plt.figure(figsize=(15, 25))
dendro = sch.dendrogram(Z_single, orientation='left', leaf_rotation=0, leaf_font_size=15)
plt.show()
plt.figure(figsize=(15, 25))
dendro_complete = sch.dendrogram(Z_complete, orientation='left', leaf_rotation=0, leaf_font_size=15)
plt.show()
```


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00


7
6
5
4
3
2
1
0

We can see that the dendrogram for complete method that group are more equally distrubuted than for single. Furthermore, we can see some single point group for the single method (for example the number 98 which is on the bottom of the first dendrogram).

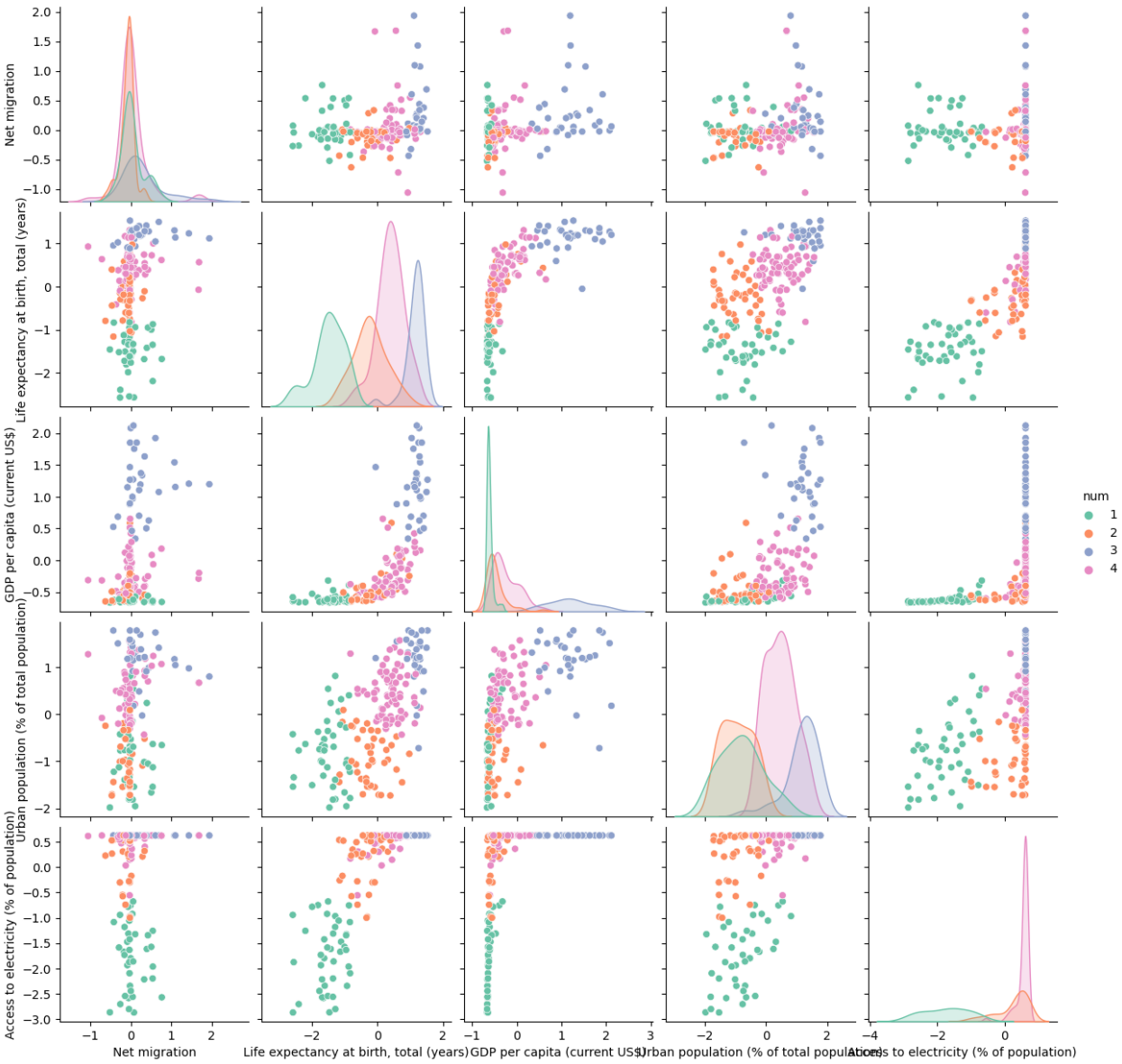
```
fig = plt.figure(figsize=(10, 7))
plt.plot(Z_complete[:,2],'o-',label="Complete method",
        linewidth=1,
        markersize=3,
        alpha=0.5)
plt.plot(Z_single[:,2],'o-',color="red",label="Single method",
        linewidth=1,
        markersize=3,
        alpha=0.5)
plt.xlabel("Number of clusters")
plt.ylabel("Distance")
plt.title("Grouping distance curve")
plt.legend()
plt.grid(axis='y')
```



The graph represents the grouping distance between the branchs of the dendrogramme. The larger the distance, the easier it is to seperate the clusters. The distance is larger with the complete method, that is why we will keep it until the end of the study. This graph also shows that a number of clusters of 5 would be efficient for this dataset.

```
clea_dat = clean_data.drop(["cluster_number"],axis=1).copy()
clea_dat["num"]=clusters_hc
clea_dat
sns.pairplot(clea_dat, hue="num", palette="Set2")
```

```
<seaborn.axisgrid.PairGrid at 0x132f6d858d0>
```

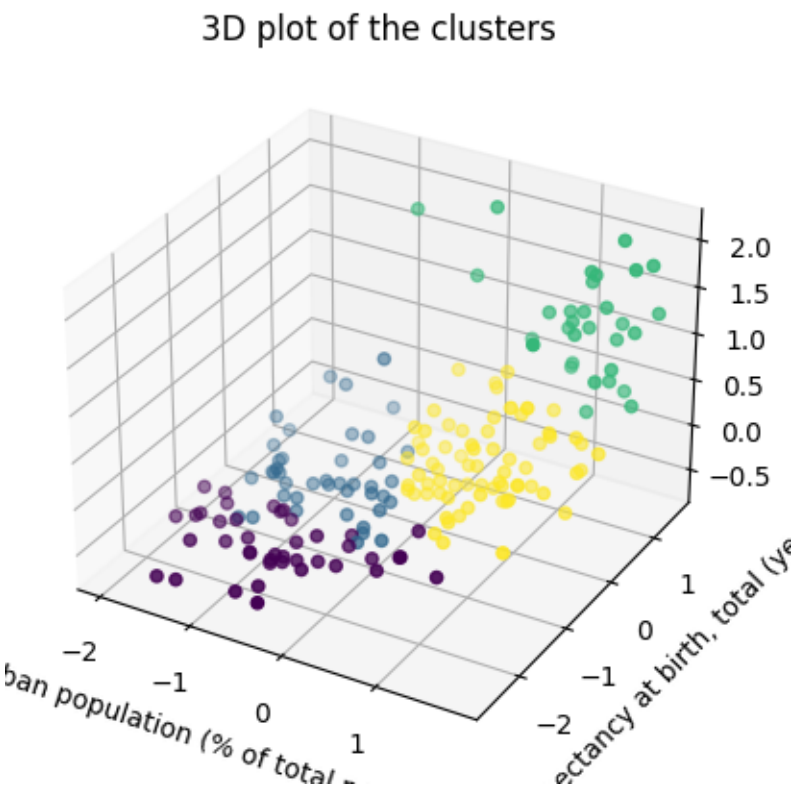


Observing the projection of life expectancy and urban population, the clusters seem to be similar to the results of Kmeans. It will be checked with the value of entropy.

3D Plot Hierarchichal clusterings

```
def plot_3d(df, clusters, axe1, axe2, axe3):
    fig=plt.figure()
    ax=fig.add_subplot(111,projection="3d")
    x=df[axe1]
    y=df[axe2]
    z=df[axe3]
    cmap = ListedColormap(sns.color_palette("husl", 256).as_hex())
    ax.scatter(x,y,z,c=clusters)
    plt.xlabel(axe1)
    plt.ylabel(axe2)
    plt.title("3D plot of the clusters")
    plt.show()
```

```
axe1="Urban population (% of total population)"
axe2="Life expectancy at birth, total (years)"
axe3="GDP per capita (current US$)"
plot_3d(clean_data,clusters_hc,axe1,axe2,axe3)
```



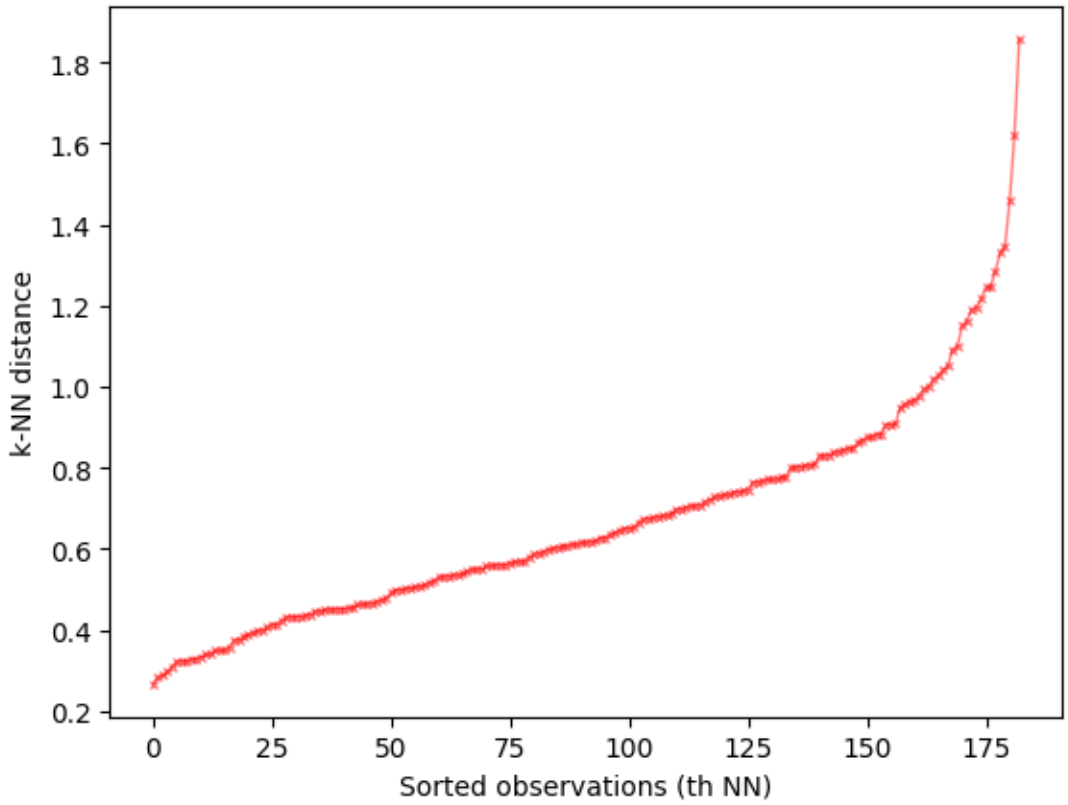
We can see more clearly the proximity of intra-clusters points in space, as well as the points that seem to be further from the center of the clusters.

DBSCAN

DBSCAN is an unsupervised method, relying on mainly 2 parameters, epsilon and min_samples. We try to estimate the optimal values, to have an efficient clustering.

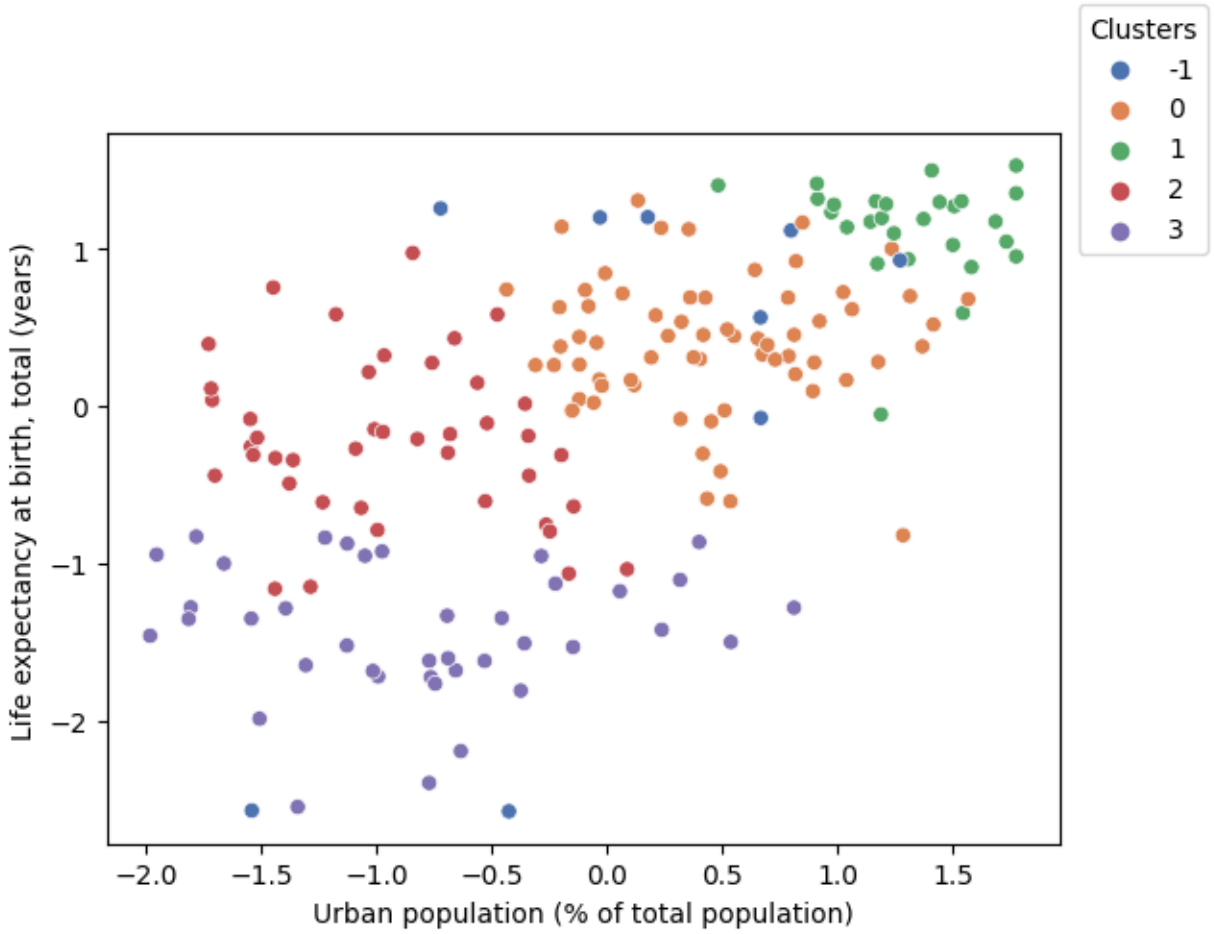
```
importlib.reload(cf)
cf.plot_knn_dist(clean_data,10)
```

```
c:\Users\supqu\OneDrive - INSA Lyon\Bureau\iaia\IA_project\clustering_function.py:67:
UserWarning: color is redundantly defined by the 'color' keyword argument
and the fmt string "bx-" (-> color='b'). The keyword argument will take precedence.
  plt.plot(
```



To estimate the value of eps to use in the DBSCAN clustering, we compute the sorted k-NN distance. The optimal value of eps should be where there is a 'elbow' in curve. Indeed, points having a value superior to eps will be considered as noise (noted in clusters -1 by DBSCAN). Here we find eps equal to around 1. The value of min_samples should be at least greater than 2*number of dimensions (= 5).

```
clusters_db=cf.dbscan_clustering(clean_data,1,12)
p = sns.scatterplot(data=clean_data, x="Urban population (% of total population)", \
y="Life expectancy at birth, total (years)", hue=clusters_db, legend="full", palette="deep")
sns.move_legend(p, "upper right", bbox_to_anchor=(1.17, 1.2), title='Clusters')
plt.show()
```



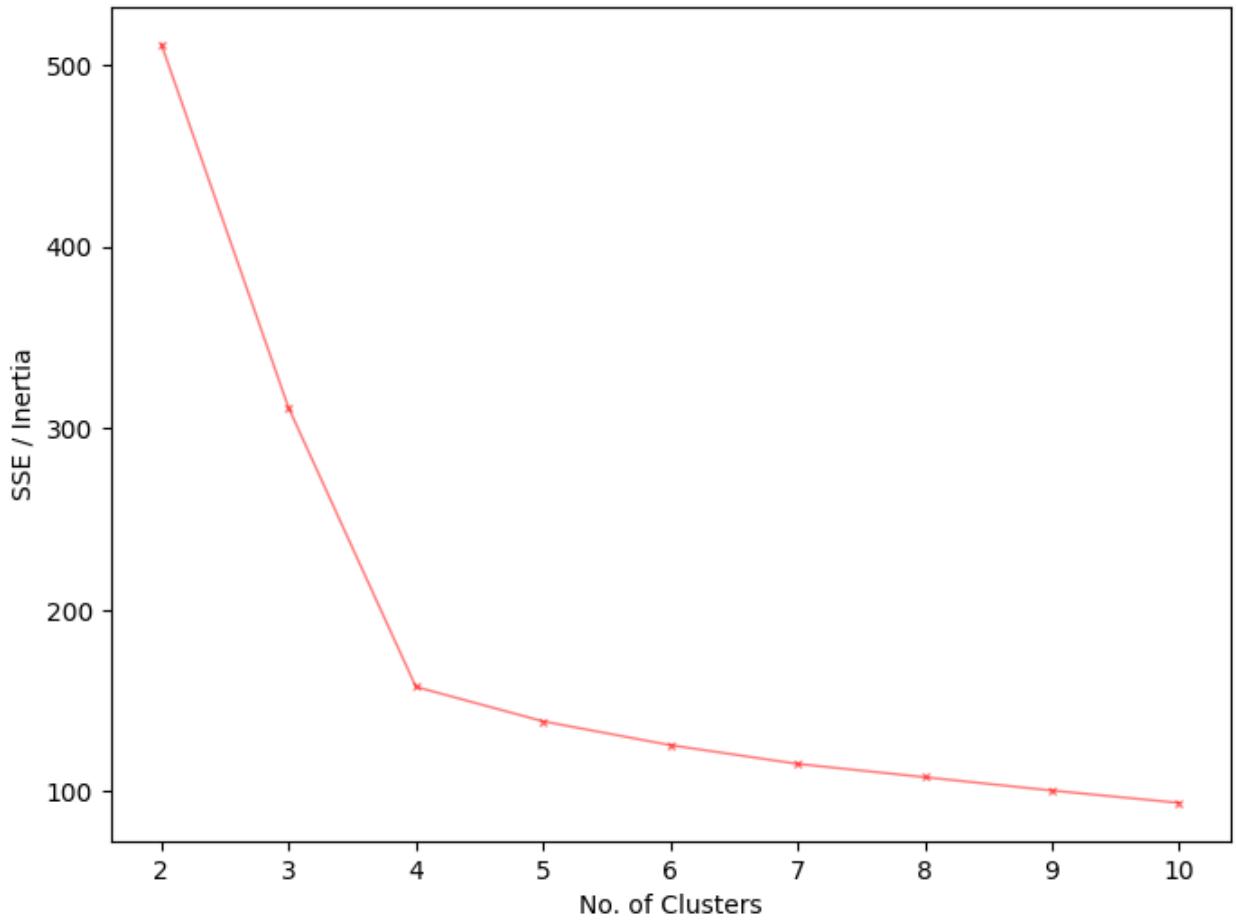
DBSCAN gives similar clusters to both previous methods: -cluster 1 corresponds to developed countries -cluster 0 corresponds to emerging countries -cluster 2 corresponds to future emerging countries -cluster 3 corresponds to under developed countries This methods also finds some noise points (in blue).

SSE Computation for Kmeans

```
importlib.reload(cf)
```

```
cf.drawSSEPlotForKMeans(clean_data,10)
```

```
c:\Users\supqu\OneDrive - INSA Lyon\Bureau\iaia\IA_project\clustering_function.py:97:
UserWarning: color is redundantly defined by the 'color' keyword argument
and the fmt string "bx-" (-> color='b'). The keyword argument will take precedence.
  plt.plot(
```



We want to minimize the SSE while having the minimal number of clusters. There is an elbow in the curve at the value of 4 clusters, which seems to be the optimal value for the number of clusters.

Silhouette scores computation for Kmeans

```
importlib.reload(cf)
cf.silhouette_score_plot(clean_data,10,20)
```

```
c:\Users\supqu\OneDrive - INSA Lyon\Bureau\iaia\IA_project\clustering_function.py:122:
UserWarning: color is redundantly defined by the 'color' keyword argument
and the fmt string "bx-" (-> color='b'). The keyword argument will take precedence.
  plt.plot(
```


images/657dfd6657280e27bac981a81925ebaaafc72a0d.png

The silhouette scores is an indicator of how well the points fits to the shapes of the clusters, so we need to have the highest value, which is reached here with the number of clusters equal to 4. According to these 2 graphs, we finally choose 4 as the number of clusters.

Stability of KMeans

To study the stability of Kmeans, we check for n_iter iterations the value of the silhouette score and the SSE with a random initial point and without setting the seed.

```
importlib.reload(cf)

sse,sil=cf.compute_sse_and_sil(clean_data,20)
_ = plt.hist(sil)
```


```
_ = plt.hist(sse)
```

For both SSE and silhouette score, there is a most frequent value. However there still some other value, which means that it is not fully stable.

Entropy

```
importlib.reload(cf)

km_vs_hc=cf.compare_entropy(clusters_km_wo_out,clusters_hc)
print("entropy of each cluster: ", km_vs_hc)
```

```
entropy of each cluster:  ([0.0, 0.0, 0.0, 0.0], col_0   1   2   3   4
row_0
0       0  42   0   0
1       0   0  31   0
2      41   0   0   0
3       0   0   0  69)
```

```
km_vs_db=cf.compare_entropy(clusters_km_wo_out,clusters_db)
print("entropy of each cluster: ", km_vs_db)
```

```
entropy of each cluster:  ([0.0, 0.5547781633412736, 0.2811937964320427, 0.2580186686648155], col_0  -1   0
row_0
0       0   0   0  42   0
1       4   0  27   0   0
2       2   0   0   0  39
3       3  66   0   0   0)
```

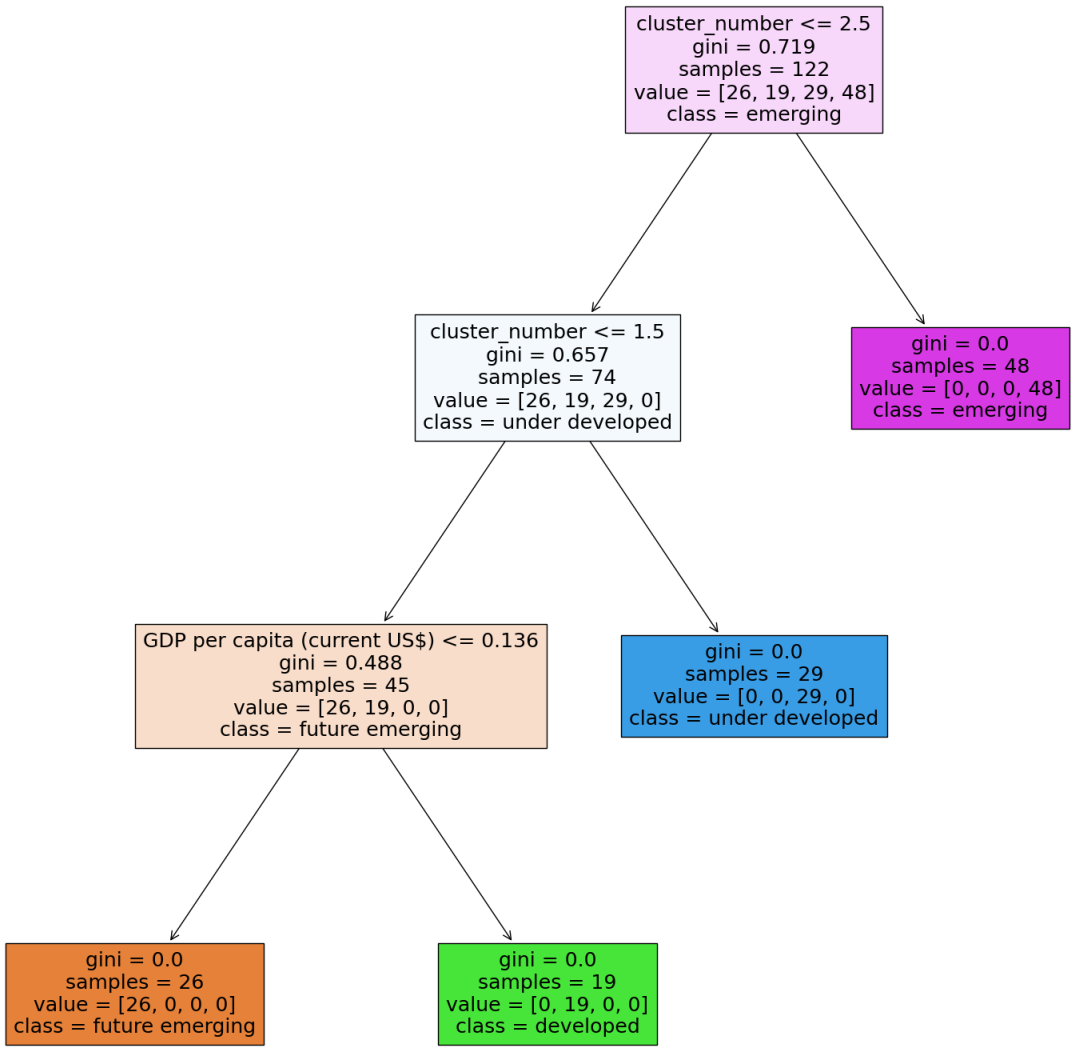
When comparing Kmeans and complete Hierarchical clustering, entropies values are equal to 0. It means that the 2 methods find the same results in our case. When comparing Kmeans and DBSCAN, 2 clusters have a entropy of around 0,2~0,3 and one of 0, which means that the results for these clusters are quite similar. However, there is a cluster with an entropy of 0,55. This can be explain by the facts that DBSCAN also have a group for noise points.

Description with decision tree

```
x_train,x_test,y_train,y_test=sklearn.model_selection.train_test_split(clean_data,clusters_km_wo_out, \
test_size=1/3, train_size=2/3)
tree=sklearn.tree.DecisionTreeClassifier()
clf = tree.fit(x_train, y_train)
y_pred = clf.predict(x_test)
```

```
f1=sklearn.metrics.f1_score(y_test, y_pred, average='weighted')
print(f1)
acc=sklearn.metrics.accuracy_score(y_test, y_pred)
print(acc)
prec=sklearn.metrics.precision_score(y_test, y_pred, average='weighted')
print(prec)
plt.figure(figsize=(20,20))
_=sklearn.tree.plot_tree(tree,filled=True,feature_names=clean_data.columns,
class_names=["future emerging","developed","under developed","emerging"])
```

```
0.983670015864622
0.9836065573770492
0.9848675914249685
```



We can see on the decision tree that the decision tree is simple but explain well the clusters. Moreover, with Kmeans clustering, the scores of f1, accuracy and precision are equal to 1.

To conclude, the Kmeans clustering is efficient to determine a relation between the attributes and the level of

development of a country. That is expected because some attributes focused on access to electricity, urban population, which are indicators of the development cities. The GDP is directly related to the economic development, and so is life expectancy to health safety. Moreover positive net immigration indicates that a country is attractive and therefore developed, and vice-versa.