Mini Project 3

Full Name: Miguel Ángel Benítez Alguacil
Student ID: 1900853

Åbo Akademi University

Machine Learning

Index

1. Introduction (problem statement)

The aim of this last project is to experiment with unsupervised learning, more precisely clustering. I need to select an indicator and perform the following tasks:

1. Cluster the indicator values for countries for your selected indicator for all available years
2. Select 3 countries which changed clusters during the years
3. Find a reason (in the same dataset) for the change. This means that you should select for each of the 3 countries another indicator whose change possibly motivated the country's cluster change.

In the next pages I will explain how I have solved the problem using Python 3 libraries and Jupyter Notebook. Also, I will talk about the following:

- Descriptive analysis of the data.
- What will I use the data for.
- How did I clean the dataset.
- Steps took to perform the task.
- Indicators.
- Clustering.
- Interpretation of the clustering.
- Conclusion.

2. Descriptive analysis of the data

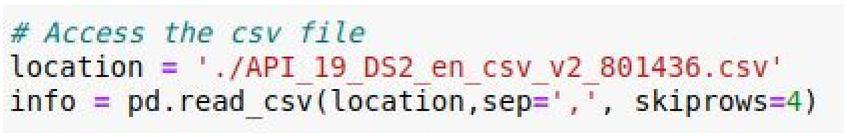
This dataset contains information on 76 economic indicators for 263 countries. Every indicator is followed during 60 years, from 1960 to 2019, depending on the indicator it has a shorter time ranges or less data. The dataset has about 20000 entries.
Some examples of these indicators are: CO2 emissions, Mortality rate in children under 5, Urban population, access to electricity, etc.

2.1. What will I use the data for

In my case, the indicator that most caught my attention was the Mortality rate in children under 5. So I will base this project in discover information about that and the possibles causes of its descent / ascent in a country.

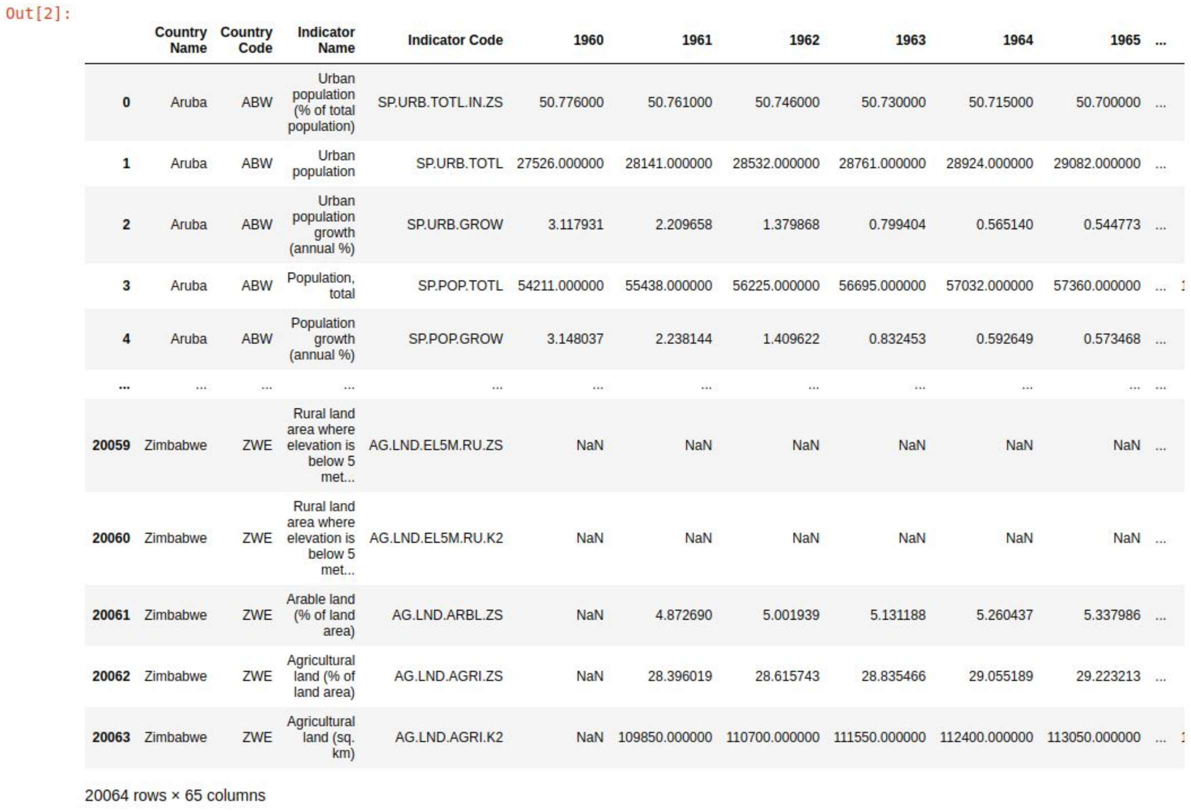
2.2. How did I clean the dataset

The first thing we need to do is to **load the data**. To do so we will use pandas, reading from the file using '**,**' as separation and **starting in the 4th row** because the document has other info that is not important in the 3 first rows.



```
# Access the csv file
location = './API_19_DS2_en_csv_v2_801436.csv'
info = pd.read_csv(location,sep=',', skiprows=4)
```

Once I had the dataset loaded, I could observe that there were a lot of **NaN** (null values) because there were cells in the excel that were **empty**.



Out[2]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...
0	Aruba	ABW	Urban population (% of total population)	SP.URB.TOTL.IN.ZS	50.776000	50.761000	50.746000	50.730000	50.715000	50.700000	...
1	Aruba	ABW	Urban population	SP.URB.TOTL	27526.000000	28141.000000	28532.000000	28761.000000	28924.000000	29082.000000	...
2	Aruba	ABW	Urban population growth (annual %)	SP.URB.GROW	3.117931	2.209658	1.379868	0.799404	0.565140	0.544773	...
3	Aruba	ABW	Population, total	SP.POP.TOTL	54211.000000	55438.000000	56225.000000	56695.000000	57032.000000	57360.000000	...
4	Aruba	ABW	Population growth (annual %)	SP.POP.GROW	3.148037	2.238144	1.409622	0.832453	0.592649	0.573468	...
...	...	...	...	...	...	...	...	...	...	...	...
20059	Zimbabwe	ZWE	Rural land area where elevation is below 5 met...	AG.LND.EL5M.RU.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...
20060	Zimbabwe	ZWE	Rural land area where elevation is below 5 met...	AG.LND.EL5M.RU.K2	NaN	NaN	NaN	NaN	NaN	NaN	...
20061	Zimbabwe	ZWE	Arable land (% of land area)	AG.LND.ARBL.ZS	NaN	4.872690	5.001939	5.131188	5.260437	5.337986	...
20062	Zimbabwe	ZWE	Agricultural land (% of land area)	AG.LND.AGRI.ZS	NaN	28.396019	28.615743	28.835466	29.055189	29.223213	...
20063	Zimbabwe	ZWE	Agricultural land (sq. km)	AG.LND.AGRI.K2	NaN	109850.000000	110700.000000	111550.000000	112400.000000	113050.000000	...

20064 rows × 65 columns

If we do **info.isnull().sum()** we obtain the number of cells that are null in each column:

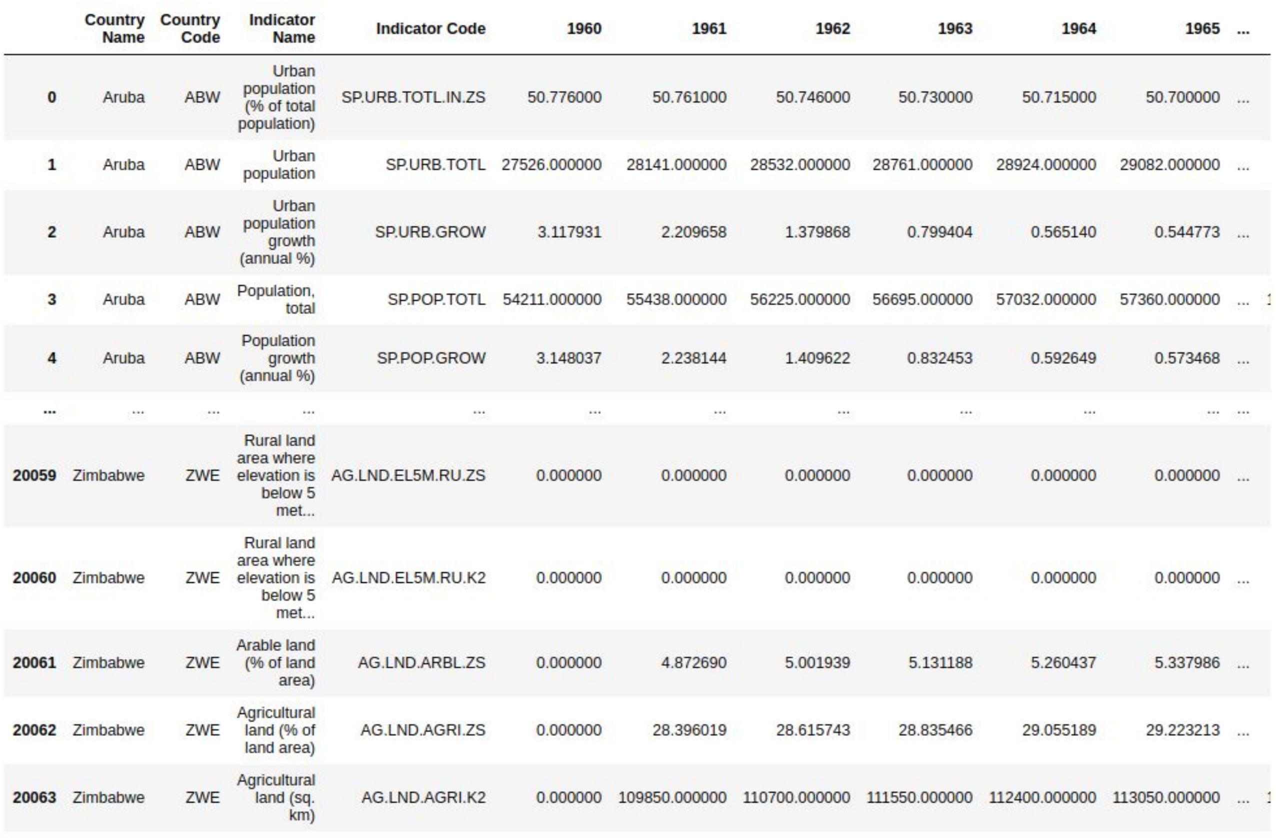
```
In [3]: info.isnull().sum()

Out[3]: Country Name          0
        Country Code          0
        Indicator Name        0
        Indicator Code        0
        1960              16607
                          ...
        2016              14842
        2017              16223
        2018              16892
        2019              19874
        Unnamed: 64       20064
        Length: 65, dtype: int64
```

To successfully complete the task I had to clean the dataset **deleting** the **last column**, changing the **values** of the **empty cells to 0**.

```
# Replace NaN with 0
for i in info.columns:
    info[i].fillna(0, inplace=True)
```

```
# The "Unnamed: 64" column has no data, so let's get ride of it.
info = info.drop(columns='Unnamed: 64')
```

This is the final result of the **cleaned dataset**:



	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...
0	Aruba	ABW	Urban population (% of total population)	SP.URB.TOTL.IN.ZS	50.776000	50.761000	50.746000	50.730000	50.715000	50.700000	...
1	Aruba	ABW	Urban population	SP.URB.TOTL	27526.000000	28141.000000	28532.000000	28761.000000	28924.000000	29082.000000	...
2	Aruba	ABW	Urban population growth (annual %)	SP.URB.GROW	3.117931	2.209658	1.379868	0.799404	0.565140	0.544773	...
3	Aruba	ABW	Population, total	SP.POP.TOTL	54211.000000	55438.000000	56225.000000	56695.000000	57032.000000	57360.000000	...
4	Aruba	ABW	Population growth (annual %)	SP.POP.GROW	3.148037	2.238144	1.409622	0.832453	0.592649	0.573468	...
...	...	...	...	...	...	...	...	...	...	...	...
20059	Zimbabwe	ZWE	Rural land area where elevation is below 5 met...	AG.LND.EL5M.RU.ZS	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
20060	Zimbabwe	ZWE	Rural land area where elevation is below 5 met...	AG.LND.EL5M.RU.K2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
20061	Zimbabwe	ZWE	Arable land (% of land area)	AG.LND.ARBL.ZS	0.000000	4.872690	5.001939	5.131188	5.260437	5.337986	...
20062	Zimbabwe	ZWE	Agricultural land (% of land area)	AG.LND.AGRI.ZS	0.000000	28.396019	28.615743	28.835466	29.055189	29.223213	...
20063	Zimbabwe	ZWE	Agricultural land (sq. km)	AG.LND.AGRI.K2	0.000000	109850.000000	110700.000000	111550.000000	112400.000000	113050.000000	...

20064 rows × 64 columns

3. Steps to perform the task.

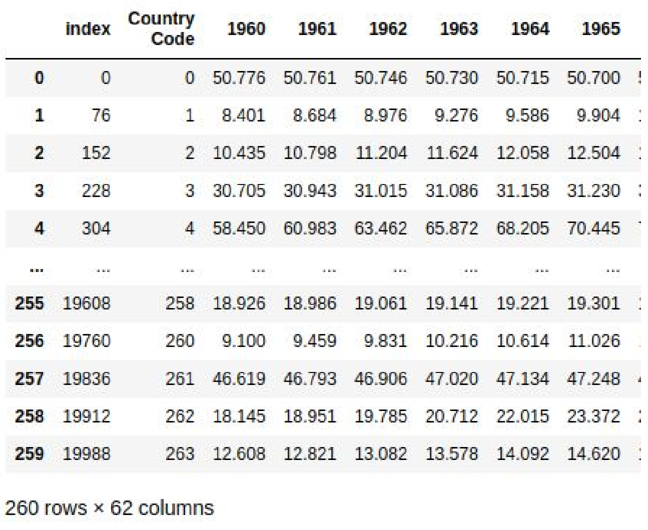
Once we have cleaned the data, we can start working with it.
As I said before, I would like to know more about the infant **mortality ratio**, and another indicator that I could use to contrast would be the **total percentage of the population**. In this way, you could see the mortality ratio taking into account the percentage of the population of the specific country.

The first thing I did was to create two subsets to get rid of unnecessary information, so I could focus on the two indicators that interest me.
The steps that I took to get a good subset were:

- **Copy** the dataset
- **Delete** all the **rows except from** the ones regarding to **the indicator**
- **Delete** all the **columns** that were not useful anymore: **'Country Name', 'Indicator Name' and 'Indicator Code'**
- **Convert** the **'Country Code'** values **into float** because we can not work with strings.
- **Delete** the **empty rows**, the countries with no data, to do so I check if they have information in **'2018'** (It is not entirely reliable but looking over the data set, the majority who did not have data had that cell empty)

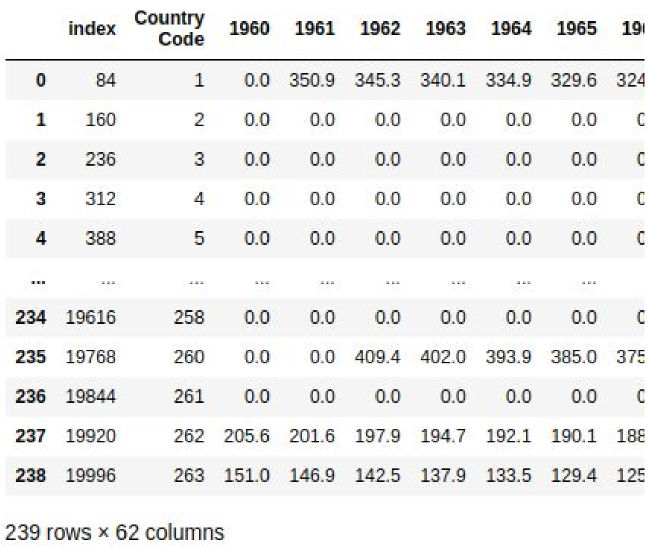
The final results were:

- The **urban_popu** dataset in which there were information about the **Urban Population (% of total population)** and had **260 rows**.
- The **mortality_rate** with information about the **Mortality rate, under-5 (per 1,000 live births)** and had **239 rows**.



	index	Country Code	1960	1961	1962	1963	1964	1965
0	0	0	50.776	50.761	50.746	50.730	50.715	50.700
1	76	1	8.401	8.684	8.976	9.276	9.586	9.904
2	152	2	10.435	10.798	11.204	11.624	12.058	12.504
3	228	3	30.705	30.943	31.015	31.086	31.158	31.230
4	304	4	58.450	60.983	63.462	65.872	68.205	70.445
...	...	...	...	...	...	...	...	...
255	19608	258	18.926	18.986	19.061	19.141	19.221	19.301
256	19760	260	9.100	9.459	9.831	10.216	10.614	11.026
257	19836	261	46.619	46.793	46.906	47.020	47.134	47.248
258	19912	262	18.145	18.951	19.785	20.712	22.015	23.372
259	19988	263	12.608	12.821	13.082	13.578	14.092	14.620

260 rows × 62 columns



	index	Country Code	1960	1961	1962	1963	1964	1965
0	84	1	0.0	350.9	345.3	340.1	334.9	329.6
1	160	2	0.0	0.0	0.0	0.0	0.0	0.0
2	236	3	0.0	0.0	0.0	0.0	0.0	0.0
3	312	4	0.0	0.0	0.0	0.0	0.0	0.0
4	388	5	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
234	19616	258	0.0	0.0	0.0	0.0	0.0	0.0
235	19768	260	0.0	0.0	409.4	402.0	393.9	385.0
236	19844	261	0.0	0.0	0.0	0.0	0.0	0.0
237	19920	262	205.6	201.6	197.9	194.7	192.1	190.1
238	19996	263	151.0	146.9	142.5	137.9	133.5	129.4

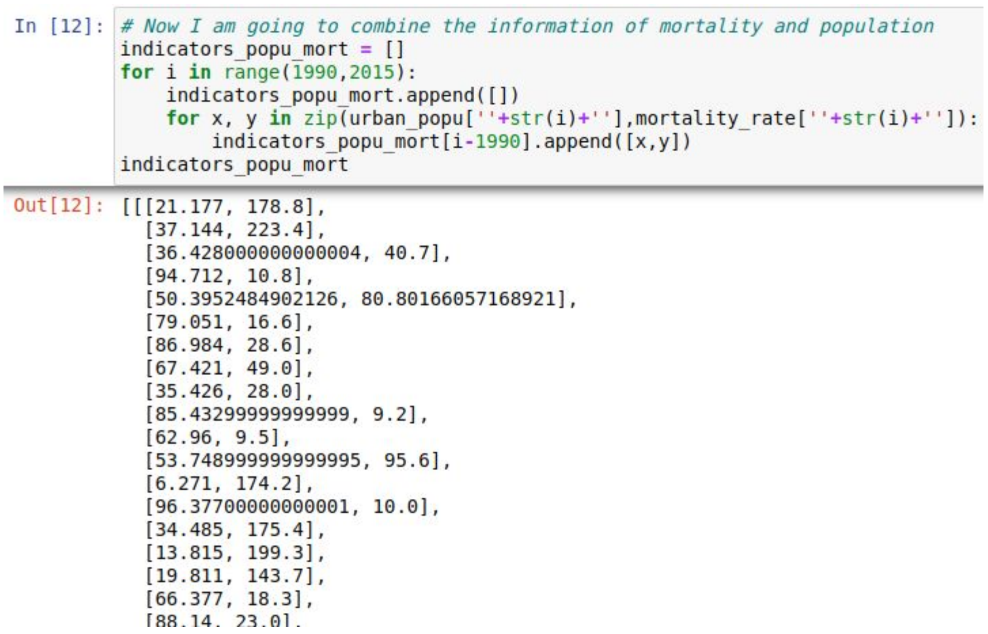
239 rows × 62 columns

As we can see, they **do not have the same number of rows**, so we will have to fix that.
After having the **same countries (238) in both sub datasets** we can combine them to **get** the **indicators**..

3.1. Indicators

Before I had experimenting also with **Energy Consumption and CO2 emissions** but I did not find them as interesting as the Mortality rate so that is why in the end I chose that one. Also I tried with **poverty** and **mortality** but there were **not enough data**, that is why I changed it for **Urban Population**.

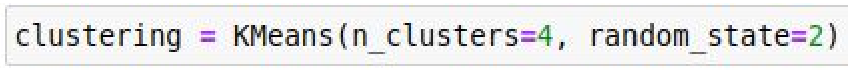
Once we have both indicators ready, we can combine them



```
In [12]: # Now I am going to combine the information of mortality and population
         indicators_popu_mort = []
         for i in range(1990,2015):
             indicators_popu_mort.append([])
             for x, y in zip(urban_popu[''+str(i)+''],mortality_rate[''+str(i)+'']):
                 indicators_popu_mort[i-1990].append([x,y])
         indicators_popu_mort

Out[12]: [[[21.177, 178.8],
           [37.144, 223.4],
           [36.428000000000004, 40.7],
           [94.712, 10.8],
           [50.3952484902126, 80.80166057168921],
           [79.051, 16.6],
           [86.984, 28.6],
           [67.421, 49.0],
           [35.426, 28.0],
           [85.43299999999999, 9.2],
           [62.96, 9.5],
           [53.748999999999995, 95.6],
           [6.271, 174.2],
           [96.37700000000001, 10.0],
           [34.485, 175.4],
           [13.815, 199.3],
           [19.811, 143.7],
           [66.377, 18.3],
           [88.14, 23.0],
```

3.2. Clustering

I have used the K-Means algorithm with 4 clusters and a random state equal to 2 (this is only to get the same values in every execution).



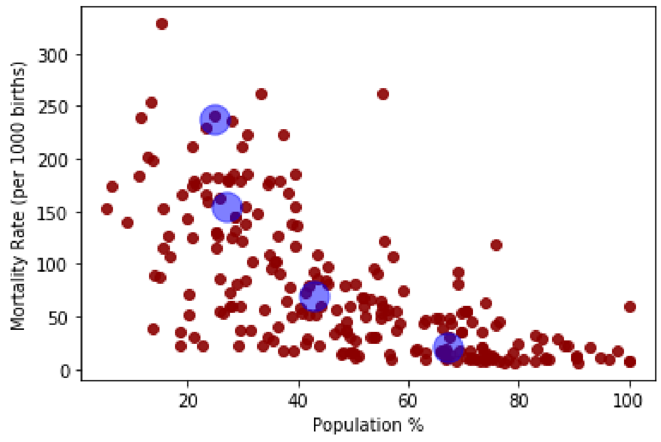
```
clustering = KMeans(n_clusters=4, random_state=2)
```

I have experimented also with 3 clusters but the data were more clear with 4.

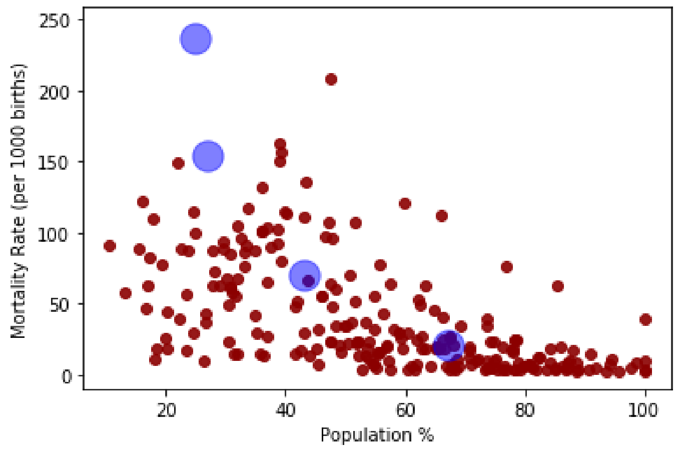
I choose to get an output for **1990** and **2010**, so we can see the evolution of the data in 20 years.

4. Interpreting the clustering.

The output for 1990 is the following:



The output for 2010 is this one:



We can observe that the **mortality rate has decreased** and also that in **2010** there are **more countries** with a very **low mortality** rage and a **huge population %**.

The values for the clustering are:

```
array([[ 42.97931532,  69.90466071],
       [ 26.91946733, 154.22581563],
       [ 67.19945615,  21.2270059 ],
       [ 25.00907143, 237.34285714]])
```

As we can see, the **4 clusters** (blue dots) represents **4 different groups of data**:

- Cluster 0: 42.97% of the population with 69.90 mortality rate per 1000 births.
- Cluster 1 and cluster 3 have the same population percentage but the mortality rate is almost 100 points different.
- Cluster 2 has the highest population percentage and the lowest mortality rate.

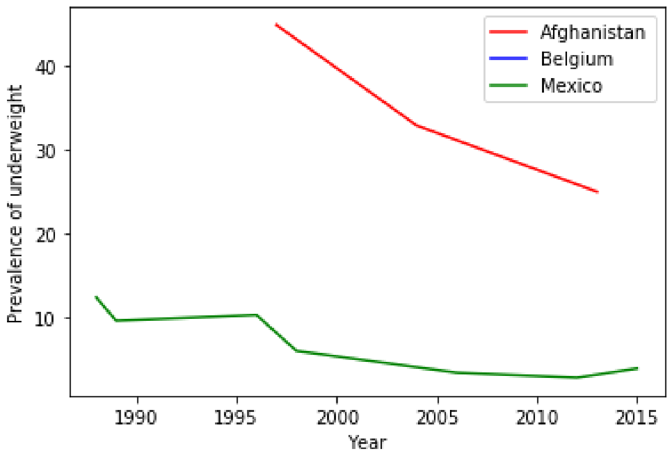
So my question at this moment was, how can I know **which countries have changed its cluster**?, for example **from cluster 3 to 1,** which would mean that they could **solved a bit their mortality problem** with the **same** amount of **population**.

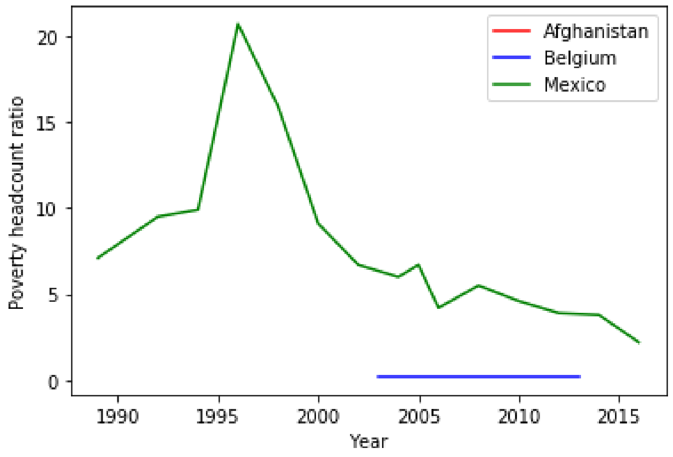
I used the **output1990** and **output2010** to compare if in **1990 they were part of cluster 3** and in **2020** they were in **cluster 1**.

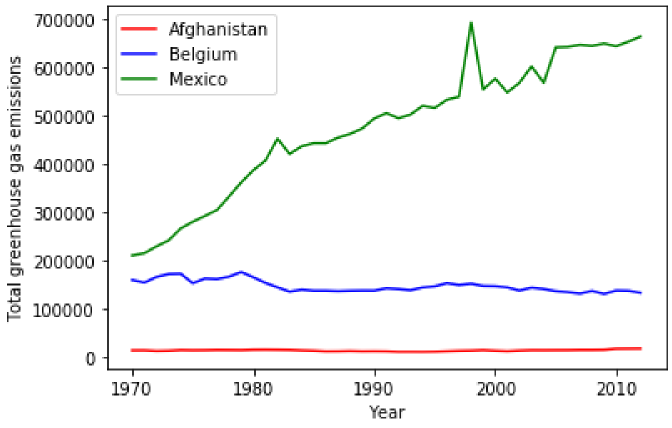
I discovered that there were **14 countries** in which this had occurred:
['Afghanistan', 'Belgium', 'Europe & Central Asia', 'Fiji', 'Faroe Islands', 'Jamaica', 'Lesotho', 'St. Martin (French part)', 'Moldova', 'Mexico', 'Marshall Islands', 'Papua New Guinea', 'Paraguay', 'Sudan']

To try to discover why this cluster change, I focused on 3 countries: **Afghanistan** (with a **low income**), **Belgium** (with a **high income**) and **Mexico** (**Upper middle income**). The other indicators I have chosen are:
- Prevalence of underweight (% of children under 5)
- Poverty headcount ratio (% of population)
- Total greenhouse gas emissions (kt of CO2 equivalent)

(To have better and clearer graphs I only plot the data that was not 0)







If we look at the **first two graphs**, we can see that the **values tend to get smaller over time** in both countries (in the first we have no data from Belgium and in the second the same for Afghanistan). However, in the **third graph** this **is not true**, I think it is because the emission of **greenhouse gases does not have as much relation to infant mortality as poverty or the prevalence of underweight** can be.

5. Conclusion.

I found this last project really interesting. I really enjoyed trying to figure it out the data and how could I compare one indicator with another, also how the indicators can affect others.

I think I have learned a lot about clustering thanks to this project, and also python.