Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Bachelor's diploma thesis

in the field of study Data Science

Application for the analysis of the economic growth indexes for European countries

Agata Makarewicz

student record book number 298827

Jacek Wiśniewski

student record book number 298849

thesis supervisor

Agnieszka Jastrzębska, Ph.D. Eng.

WARSAW 2022

....

supervisor’s signature author’s signature

Abstract

Application for the analysis of the economic growth indexes for European countries

Clustering of European countries has been a popular topic among economists for over 30 years. Most of the attempts at creating reliable divisions of these countries resulted in creating 2 to 5 groups. We propose a new clustering attempt, which takes into account the time series of economic growth and divides countries into groups using three different methods: K-means clustering, Agglomerative clustering, and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The results compare Euclidean and Dynamic Time Warping (DTW) similarity measures used in clustering models and verify previously proposed divisions.
Keywords: economics, clustering, business cycles, Dynamic Time Warping, K-means clustering, Agglomerative clustering, Density-Based Spatial Clustering of Applications with Noise

Streszczenie

Aplikacja do analizy wskaźników wzrostu gospodarczego państw europejskich

Klasteryzacja państw europejskich jest tematem często poruszanym przez ekonomistów przez ostatnie 30 lat. Większość prac naukowych związanych z tym problemem tworzy od 2 do 5 grup państw. W tej pracy zaprezentowane jest nowe podejście do tematu, które wykorzystuje szeregi czasowe indeksów ekonomicznych do przeprowadzenia klasteryzacji trzema algorytmami: klasteryzacja K-średnich, klasteryzacja hierarchiczna oraz klasteryzacja bazująca na gęstości (DBSCAN). Dodatkowo zostały przetestowane dwie miary podobieństwa na wspomnianych algorytmach, miara euklidesowa i algorytm DTW, a wyniki klasteryzacji zostały porównane z podziałami wykonanymi we wcześniejszych pracach naukowych.
Słowa kluczowe: ekonomia, klasteryzacja, cykle biznesowe, DTW, DBSAN, K-średnich, klasteryzacja hierarchiczna

Warsaw, ...

Declaration

I hereby declare that the thesis entitled „Application for the analysis of the economic growth indexes for European countries", submitted for the Engineer degree, supervised by Agnieszka Jastrzębska, Ph.D. Eng., is entirely my original work apart from the recognized reference.

....

Contents

1. Introduction

1.1. Problem description and Motivation

For over 30 years, scientific works have presented various divisions of European countries into economic and cultural groups, based on different criteria such as GDP per capita, level of industrialization, or HDI. Depending on the considered indicators and the date of the analysis, usually, 2 to 5 groups are defined. For instance, in the article written by C. Gräbner et al. (2019), the central, peripheral, and Eastern European countries, as well as financial centers, were distinguished. Furthermore, in W. Bartlett and I. Prica's thesis (2016), there were five country groups exposed: Inner Core, Outer Core, Inner Periphery, Outer Periphery, and Super Periphery.

In this paper, we apply several standard clustering methods such as K-means, Agglomerative clustering, and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to time series of economic growth to group countries and verify the previously proposed divisions. The algorithms are evaluated using the existing cluster analysis assessment indexes, e.g. silhouette score, Dunn index, and Calinski-Harabasz index. The thesis is based on publicly available data, including the Penn World Table.

The data preprocessing phase is a significant part of the project as well. Chapter 3 describes all the steps taken to perform reliable clustering, e.g. missing value imputation, outlier detection, or data normalization. To achieve the best possible clustering results, we test multiple preprocessing options and compare their results in chapter 4, for instance, different similarity measures and different data transformations.

The implemented models are part of the web application in which the user can compare the results of the methods used, select variables and parameters for the models, as well as the development indicators presented in the charts. Visualizations of the clusters obtained with different clustering methods are also available.

1.2. Related work

Clustering of European countries has been a popular topic among economists for over 30 years. The 'core-periphery' division is often the starting point of research, however, as mentioned by Belke et al. (2016) [2], there is not any clear definition of which country belongs to the periphery and which to the core group. Therefore, every attempt at clustering European countries has its own, slightly different approach, which is described in Markus Ahlborn and Marcus Wortmann's (2018) thesis [1]. For instance, Bayoumi and Eichengreen's (1993) [3] paper uses German business cycles as the reference for the core group, thesis written by Pentecôte and Huchet-Bourdon (2012) [4] finds France as a better 'anchor point' for core countries, and the study by Aguiar-Conraria et al. (2013) [5] suggests dividing the core group into Germany group and French group.

In this thesis, we do not assume any division a priori. Instead, we compare our results to the papers mentioned above, adding a new perspective to the discussion about European countries clustering.

1.3. Division of work

Table 1.1: Division of work

Name	Responsibilities
Makarewicz	data pre-processing, implementation of clustering methods, implementation of the graphical user interface, interpretation of the results
Wiśniewski	collecting data and exploratory data analysis, comparison and analysis of development indicators and clustering results, implementation of the web application backend, interpretation of the results

2. Solution proposal

2.1. Description

The first step of the project is the overview of open-access data such as World Bank or Eurostat databases, in search of indicators which are useful in business cycles identification. After choosing potentially relevant data, datasets are downloaded to predefined directories, in Excel or CSV format, with a modified filename if needed. The next step is to explore the collected data, verify its quality, and perform the necessary preprocessing. To this end, the Jupyter Notebook file is prepared. Firstly, datasets are loaded using the Pandas library and represented by data frames. Secondly, variables statistics, their distribution, missing values and correlations are analyzed in order to identify variables which need to be dropped. In parallel with this process, new variables are constructed to deal with high correlation cases. After data integration and final selection, necessary preprocessing is performed to rectify quality issues. This process includes mainly imputation of missing values, data normalization, anomaly detection and removal. All mentioned operations are performed within a single Jupyter Notebook, after running which final dataset in CSV format is returned. The next step of the project is to apply various clustering algorithms to the chosen and pre-processed time series of economic growth, in order to group European countries and verify the previously proposed divisions. Models will be implemented in a Python script, using the `scikit-learn` and `tslearn` libraries. As an input file, the dataset returned by the previously mentioned notebook is taken. There are to be three different clustering methods implemented: k-means, hierarchical clustering and DBSCAN. Those algorithms will be evaluated using the existing cluster analysis assessment indexes - silhouette score, Calinski-Harabasz index and Dunn index – to verify the quality of performed grouping and homogeneity of obtained clusters. The analysis will cover complete time series and selected segments. The implemented models and their evaluation will be part of the web application with a graphical user interface written in Django. It will present the results of the work, allow user to compare the results of the methods used, select variables and parameters for the models, as well as the development indicators presented in the charts. Visualizations of the clusters obtained with different clustering methods will also be available.

2.2. Technology selection

The majority of this thesis focuses on data preprocessing which is fully written in Python language. For data transformation, clustering, and data visualization, we use the following Python packages and libraries:

- Numpy
- Pandas
- SciPy
- Scikit-Learn
- Matplotlib
- Plotly
- Dtaidistance
- pycountry
- skfda
- tslearn

Furthermore, to measure the quality of clustering, there are imported two R libraries using the rpy2 python package: clValid and symbolicDA. Finally, clustering model results are presented in the Django application. During the project, we used Windows as an operating system and Github as version control.

3. Data understanding and preparation

3.1. Data collection

Some of the widest and most popular sources of open-access data are World Bank and OECD databases, and when it comes to Europe, also Eurostat. In terms of economic data, however, Penn World Table is the most established data source, and therefore suitable for this project. Checking the data using the online viewer available on the webpage shows that only a few indicators important in business cycles identification are missing. These indicators, such as inflation and unemployment are to be found in World Bank Open Data, or other mentioned sources. However, one of the crucial indexes – the Human Development Index – is only available directly on the webpage dedicated to the report in which it is published (Human Development Report). Given the topic of this thesis, as well as the amount and quality of relevant data offered by different open sources, the three following data sources have been chosen:

- Penn World Table – a database with indicators on relative levels of income, capital, employment, national accounts, population and productivity, covering 183 countries between 1950 and 2019 (version 10.0). It is developed and maintained by researchers from the University of California, Davis and the Groningen Growth Development Centre of the University of Groningen.

- World Bank Open Data – a collection of databases developed and maintained by Development Data Group of World Bank Group, containing indicators on a variety of topics, including health, climate, education, economic sectors, and more. Data mainly comes from World Bank Group surveys and data collection efforts, other international organizations such as UN specialized agencies, or the statistical systems of member countries.

- Human Development Reports – annual reports published by the Human Development Report Office of the United Nations Development Programme (UNDP). They have been released since 1990, exploring different themes through the human development approach and publishing one of the key development indicators – the Human Development Index.

OECD and Eurostat databases have also been considered, but it turned out that World Bank Open Data offered the same information (indicators) for more countries and longer periods.

Loading the chosen data requires a brief look at the raw Excel/CSV files to understand their structure.

- Penn World Table dataset is downloaded as an Excel file, containing data in the „Data" sheet.
- World Bank Open Data datasets are downloaded as CSV files (XML and Excel options available) within a Zipped folder alongside metadata files containing information about given indicators and countries for which its values are provided. For each indicator, files need to be extracted and data is loaded from the file with a name starting with API_SI; four rows need to be skipped.
- Human Development Index data is downloaded as a CSV file; while loading, five rows need to be skipped.

All datasets are placed in a dedicated directory in the corresponding subdirectories. Penn World Table file is loaded with no changes. In the case of World Bank data, filenames are changed to the names of the indicators to make them easier to identify. Since for each indicator from this source there is a separate file, data is loaded by iterating over the dedicated folder, and filenames become variables' names in the process. Additionally, the HDI data filename is replaced by "Human_Development_Index" to make it more user-friendly.

At the end of the collecting process, there is a place for a data overview. Basic information about the datasets is in the following list and more sophisticated exploratory analysis can be found in the next section.

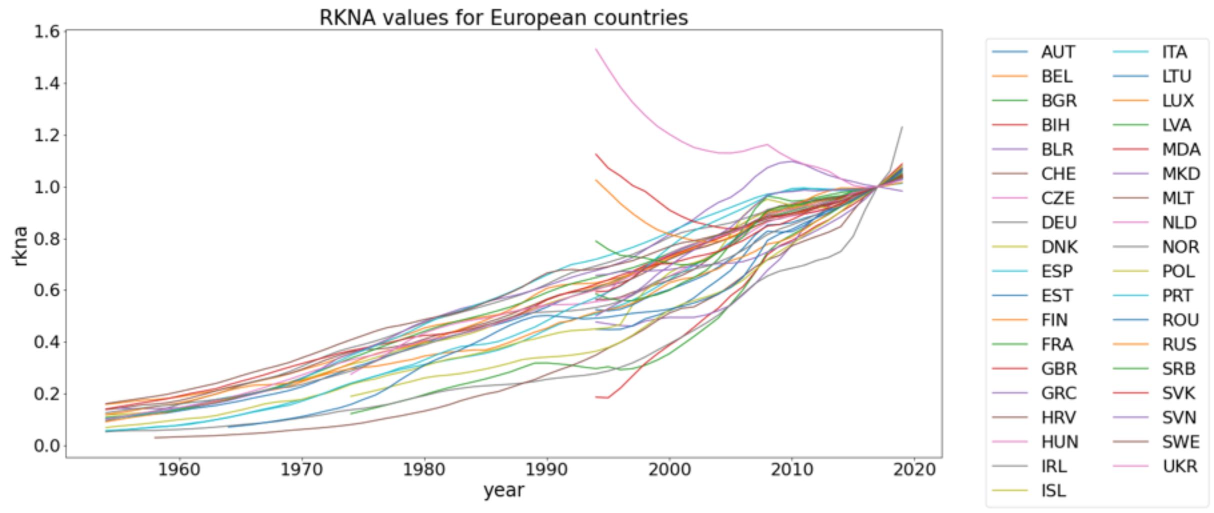
- Penn World Table is a dataset with 44 numerical columns, 8 character columns, and 12810 records. The share of missing values in the columns varies from 0
  - Identifier variables
  - Real GDP, employment and population levels
  - Current price GDP, capital and TFP
  - National accounts-based variables
  - Exchange rates and GDP price levels

  - Data information variables
  - Shares in CGDPo
  - Price levels, expenditure categories and capital

- From World Bank Open Data have been chosen 13 datasets with important indicators missing from the Penn World Table. Each of them has 4 identifier variables (country name, country ISO code, indicator name and code) and multiple numerical columns, each one corresponding to one year, with information about:
  - CO2 emission
  - Employment by economic sectors
  - Export and Import
  - Inflation
  - Migration
  - Population by age
  - Unemployment
  - Urban population
- Dataset downloaded from Human Development Reports webpage has 32 columns containing information about country name, the value of Human Development Index for every year from 1990 to 2019, and rank according to the latest HDI value. It has 206 rows, every row representing one country, region, world, or level of human development

3.2. Exploratory Data Analysis

Data described in the previous section is available for countries from all around the world, therefore to conduct insightful exploration, filtering of the European countries needs to be done at the very beginning, as they are the subject of this thesis. For that purpose provided ISO codes and country names are useful. It appears that there are 46 European countries for which any data is available. However, seven countries have any information on only five indicators, and another eight countries do not have any information on at least one indicator. These cases can lead to problems in the subsequent steps and are described in detail in the further part of the chapter.



Analyzing indicators' values in time, there are several conclusions which can be drawn by just looking at the plots:

- there are 16 indicators with relative values – either value for 2017 or USA is taken as a baseline (presented above)
- most of the indicators have a lot of missing values before 1990-1995 (presented above); it is explainable because a lot of European countries were formed or gained full independence in those years

Another step in the analysis is correlation matrix calculation to investigate the dependence between indicators. There are groups of variables present which pairwise have a very high correlation coefficient value, and they are mostly the ones from PWT. Based on the data description, these groups contain different versions (calculated differently) of one indicator, for instance, GDP variations.

This preliminary analysis leads to a list of data quality issues, which need to be addressed:

- Different measurement units – datasets contain multiple indicators in different units, on different scales, for instance, the population is given in millions whereas import values in a share of GDP and HDI on scale 0-1; therefore the data needs to be normalized before further processing and modelling
- Missing values – there are multiple missing values across the analyzed datasets, mostly because some of the European countries have gained full independence (or has been formed) around 1990-1995; there is also an indicator which was not proposed until 1990 (HDI) therefore there is no previous data on it; another case is that for some countries there is no

data at all on some indicators; on top of that, all those missing values are represented by different symbols, for instance, whitespace or colon; they need to be replaced by one value (for instance NaN) to obtain consistent representation

- Different geographical entities – World Bank data contains indicators' values not only for individual countries but also for the regions (for instance South Africa, Central Europe); for those regions, there are no officially assigned ISO codes and therefore they are not recognized by Python packages; they need to be filtered out using a dictionary of countries available in one of the packages

- Improper location data – one of the datasets (HDI) does not contain a column with country codes, instead of that only countries' names are provided, however, there are leading whitespaces present and unnecessary elaboration on countries' names (for instance 'The republic of') is added, which make them unrecognizable for Python packages; such data needs to be cleaned before further processing to be able to assign ISO codes

- Different granularity – almost all indicators' values have been collected yearly, however, for one of them (Net migration) there are only five-year estimates available; such data needs to be resampled and imputed to obtain consistent granularity

3.3. Selection and data preprocessing

Before the process of selecting data, there should appear a discussion, whether all needed fields are already collected or created. Despite having multiple indexes in the datasets, there are two more that can be composed of the collected data and have a beneficial impact on further analysis:

- GDP per capita – equal to GDP value divided by population; created not to mix data sources (there is data available for this indicator, but not in Penn World Table, from which GDP and population values are taken)

- Percentage of employed – equal to the number of people engaged (employed) divided by population; created to deal with high correlation between mentioned variables

After creating new fields, there is a data integration issue to be solved. At this step the data from each source has a column countrycode containing ISO 3166-1 alpha-3 codes for the analyzed countries. Data from Penn World Table and World Bank already had it, whereas, in case of HDI data from UNDP, it was added based on the cleaned column with countries' names. All three datasets are merged on the countrycode column into one dataset.

Newly created table has over 60 indicators available. Relevant data is chosen in a few stages.

1. Nonnumerical variables are dropped, such as currency_unit, indicator_name and ones from PWT's Data information variables group.
2. Based on the data description and literature, not all indicators from PWT are important for business cycles identification – not needed ones are dropped. Moreover, exploration and again data description shows that some of the indicators have relative values (values for 2017 are denoted as 1); such variables are also dropped.
3. The correlation matrix is calculated to investigate the dependence between indicators. Based on the values of the coefficient highly correlated pairs of variables (approximately on 0.9 and higher level) are identified and one of them is dropped. For instance, PWT provides GDP values calculated in 5 different ways, all highly dependent therefore only one of them is left for further analysis.
4. Another important factor in the data selection process is investigating the number of missing values to verify variable's completeness and usability. There are 3 dimensions regarding which amount of missing data needs to be examined:
    - amount of missing values for a given indicator (variables with only 60-70% of data present, or less, are dropped; imputation on such scale would lead to bias and artificial similarity of countries)
    - amount of missing values for a given year (if there are no values on most of the indicators for a given year, it is left out of further analysis)
    - amount of missing values for a given country (as above)

    Except for the above, there is one particular situation, that needs to be considered. Some variables, regardless of the percentage of missing values, might not have any values present for a particular country. In such a case reasonable imputation is impossible, so either variable or country must be dropped. In general, the aim is to characterize as many countries as possible with a maximum number of variables. Given the task, it is less desirable to leave a country out, because the fewer countries, the less interesting analysis. However, it is important to have a sufficient number of variables to identify cycles.

Taking all those issues into account, there are 26 indicators left, collected for 39 countries, in the years 1990-2019. The last unsolved issue before creating the first clustering models is getting rid of missing values. Depending on the distribution of missing values, different methods are to be used. Situations, where countries do not have any data in one variable, are described in the previous chapters. For the rest scenarios, the recommended option is to impute data. The best imputation algorithm is to be chosen by the special research. In this research, some known data are removed and then imputed. The best algorithm is the one that will achieve the lowest mean squared error. Below there are presented results of the research.

- Some countries do not have data for the last or the first measured date. In this situation, it appeared that the best imputation algorithm is to impute the last known value. Figure 3.1 presents scenario, in which this method is used.

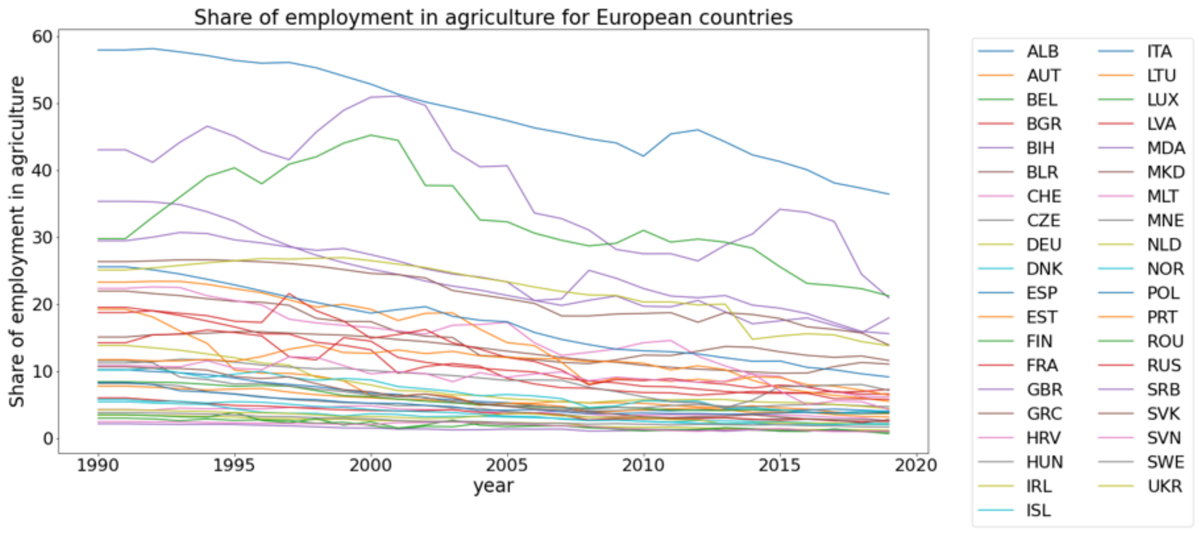


Figure 3.1: Share of employment in agriculture for European countries after missing values imputation. For this feature, data for 1990 year was missing.

- There are some indicators which started to be collected for a few countries couple of years later than for other countries. In such cases, the best option is usually to interpolate data. The only indicator that belongs to the group described above but has not been interpolated is inflation presented on Figure 3.2 and 3.3. Due to the irregular behaviour of the indicator around the 1990 year, it appeared that the best imputation option is the "K nearest neighbours" algorithm.

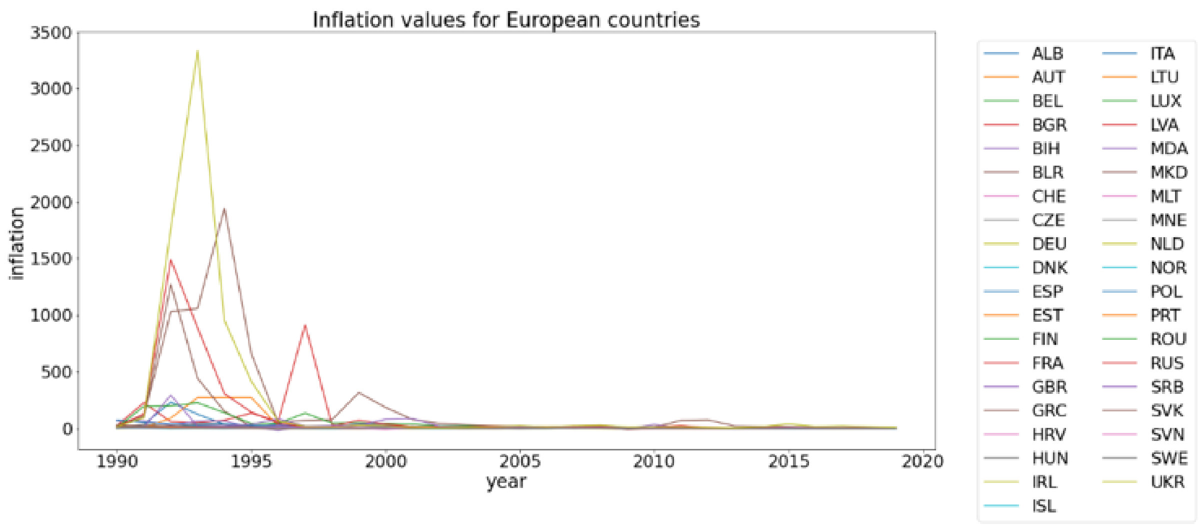


Figure 3.2: Inflation values for European countries after imputation. For this feature, 17 countries missed data for 1990-1995 period.

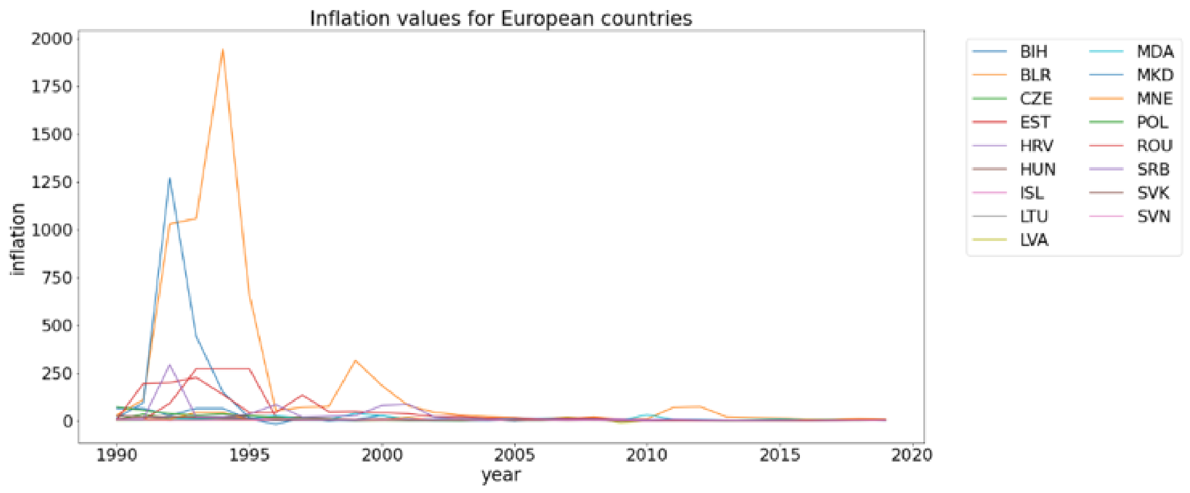


Figure 3.3: Imputed inflation values for European countries. Presented 17 countries with missing data for 1990-1995 period.

- The last indicator that needed imputation is net migrations. In this scenario, data for this indicator was collected once every 5 years. To fill the gaps there is performed linear interpolation.

Another important step is data normalization. As already mentioned in the previous section, the final dataset contains multiple indicators in different units, on different scales. Therefore data needs to be normalized before passing it to clustering models, in order to unify feature range (to $[0, 1]$) and avoid the biased contribution of indicators with greater values.

Apart from standard preprocessing methods already mentioned i.e. imputation and normalization, time series data often requires more preparation prior to being modeled with machine learning algorithms. Common approach includes using power transforms. They remove a shift from a data distribution to make it more similar to Gaussian distribution. In case of time series data, power transforms result in obtaining the effect of removing a change in variance over time. In this project, two popular power transforms are used - the log transform and its generalized version, the Box-Cox transform. Both of them are applicable only to positive values and the dataset contains some variables with negative ones. Therefore, before applying the power transforms, normalization is performed. Afterwards, data is shifted to $[1, 2]$ range (only in the function call) in order to avoid values equal to 0 and obtaining infinite results because of them. Both mentioned power transforms i.e. log transform and Box-Cox transform, are used and analyzed separately, in different data preprocessing pipelines.

Another common time series transformations are smoothing techniques. Their idea is to remove noise from the data and get a better representation of the patterns and trends it contains. In this project, `NadarayaWatsonSmoother` from Python `scikit-fda` package is used. It is an example of kernel smoothing methods, which compute the smoothed value of a given observation by taking into account the impact of each observation over it. The approach makes use of the Nadaraya-Watson estimator which can be described as a series of weighted averages, with a kernel function (by default the normal one) as a weighting function. For each point of the estimator at a given time, the peak of the kernel function is located at the same time, hence the highest weights are assigned to neighboring observations.

Last but not least, before the modeling phase, outlier detection (and treating) needs to be performed. To find abnormal indicator values in the analyzed dataset, a simple "3 times the standard deviation" rule is used. It is a heuristic method, often referred to as the three-sigma rule, which marks as an outlier every observation which absolute value exceeds 3 times the standard deviation of the data (variable). In case of detecting an outlier, its value is replaced by the mean of the previous and next values; if it is the first or last value in the series, then mean of all the values is inserted. Given the nature of analyzed data, multiple outliers will probably be explainable due to countries' structural changes. Therefore, handling outliers is deliberately not included in each preprocessing pipeline, only treated as separate approach, to analyze whether or not those observations help us distinguish countries more appropriately.

4. Model descriptions

There are multiple clustering algorithms available, ones more suitable for time series data than the others. After an overview of different methods, three algorithms were chosen, each one belonging to a different group described below.

1. **Centroid-based algorithms** organize points based on their distance from the cluster centre and aim to minimize the distances between points within one cluster. Best for dense clusters, far away from one another, assumes they are convex shaped.

2. **Hierarchical algorithms** create trees (hierarchy) of clusters. There are two approaches possible: starting with treating each point as a separate cluster and recursively merging them or treating all points as one and then dividing them instead.

3. **Density-based algorithms** detect areas of high point concentration separated by regions of the lower density of points. They are suitable for data containing outliers and clusters of arbitrary shapes.

Most of the algorithms use by default Euclidean distance as a metric. In order to extract full information from multivariate time series data, a more adequate Dynamic Time Warping (DTW) metric should be used. DTW is an algorithm for measuring similarity or distance between two sequences that may vary in time or speed. It is most useful and efficient for comparing time series data when the time indices between comparison data points do not sync up perfectly. This key feature comes from the enabled in the algorithm one-to-many and many-to-one connections which help DTW to search for similar patterns in time series with different lengths of periods. The difference between Euclidean distance and DTW metric idea is presented in Figure 4.1.

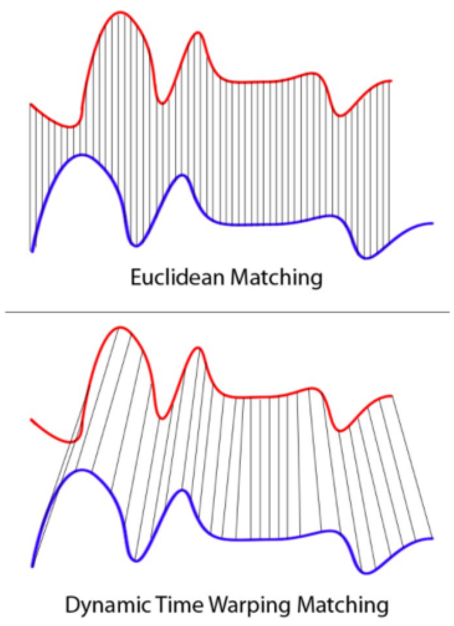


Figure 4.1: Difference between Euclidean Matching and DTW Matching

4.1. KMeans

The centroid-based algorithm used in this project is K-Means. Proposed implementation uses the TimeSeriesKMeans model from Python `tslearn` package and requires only one parameter to be passed by the user:

- n_clusters – number of clusters to form

The other important parameter, metric, is permanently set to 'dtw' to use the DTW metric to calculate the distance between points. K-Means is the most popular and the simplest method, which aims to group objects into k clusters, choosing centroids (cluster centres; "mean" of all the points in the cluster) that minimise the inertia, or within-cluster sum-of-squares criterion:

$$\sum_{i=0}^{n} \min_{\mu_j \in K}(||x_i - \mu_j||^2) \tag{4.1}$$

where K - number of clusters, n - number of samples, x – sample and μ – centroid.

Inertia measures how internally coherent the clusters are. The grouping is achieved by repeatedly relocating centroids and reassigning countries to the closest centres, which are firstly randomly initialized. To choose an optimal number of clusters, the elbow method presented in Figure 4.2 can be used.

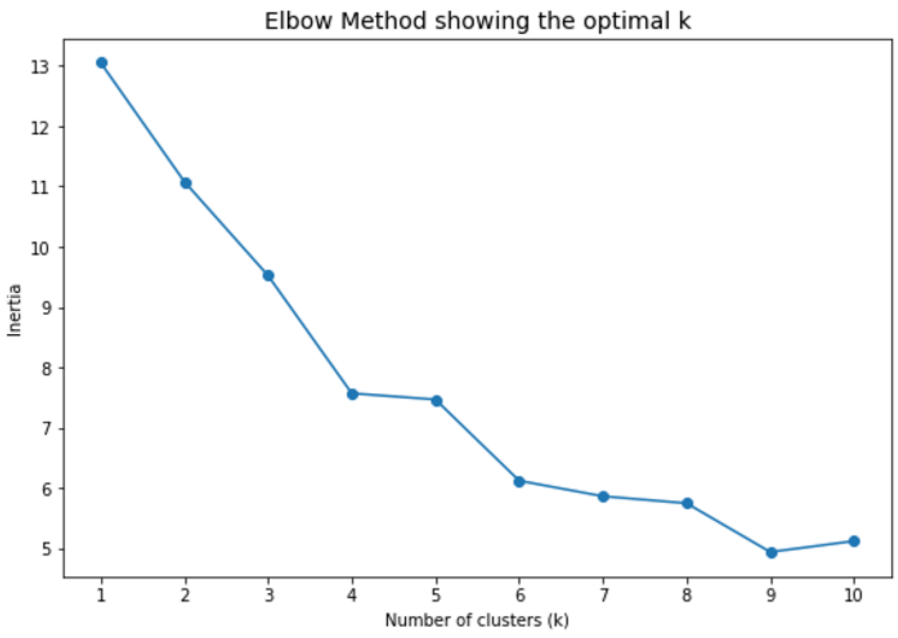


Figure 4.2: Inertia for different n_clusters parameter values, for the KMeans clustering, performed on the analyzed dataset

Figure 4.2 presents the inertia depending on the n_clusters value for the K-Means algorithm. As already mentioned, the method aims to minimize the inertia to obtain coherent clusters. One of the methods to choose optimal (yet subjective) number of clusters is the Elbow method, according to which best k is the one that relates to the point in Figure 4.2 where inertia starts to decrease slower, for instance, 4.

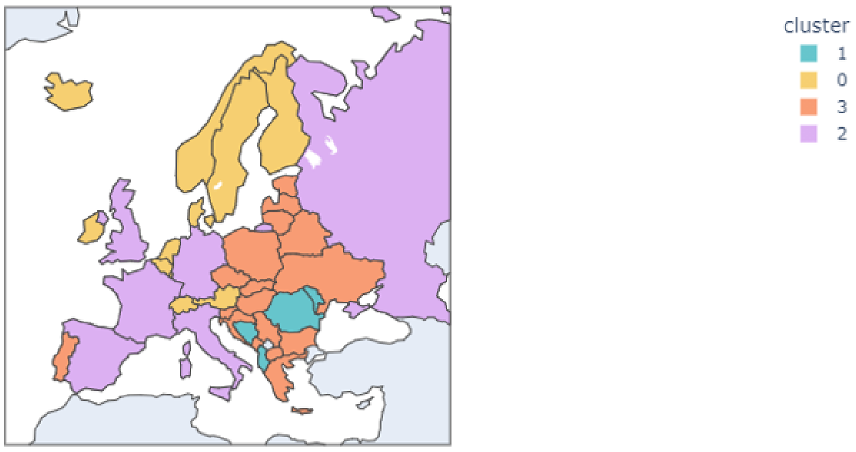


Figure 4.3: Results of K-means clustering for k=4

4.2. Agglomerative clustering

Agglomerative clustering is a hierarchical algorithm that groups the objects according to the “bottom-up” approach. Proposed implementation uses AgglomerativeClustering model from Python `scikit-learn` package and requires two parameters to be passed by the user:

- `n_clusters` – number of clusters to form
- `linkage` – type of linkage criterion, i.e. the approach to be used for computing the distance between two clusters; the pairs of clusters that minimize this criterion (are closest to each other due to chosen criterion) are merged
  - `average` – average value of all pairwise distances between the elements in the first cluster and the elements in the second cluster
  - `complete` – maximum value of all pairwise distances between the elements in the first cluster and the elements in the second cluster
  - `single` – minimum value of all pairwise distances between the elements in the first cluster and the elements in the second cluster

The default linkage criterion is usually `ward` (in used `scikit-learn` implementation as well), which minimizes the variance of the merged clusters. However, it is not compatible with the DTW metric, therefore not possible to use in this project. Another important parameter is `affinity`, which is permanently set to `precomputed`. This option means that instead of data represented by a data frame with observations and features, a previously calculated distance matrix is passed as an argument, which is because AgglomerativeClustering does not support calculating distances for multivariate time series. The distance matrix is calculated using Python `dtaidistance` package, it is square, with a dimension equal to a number of observations to group. There is also another parameter permanently set - `compute_distances=True` – which makes plotting dendrogram (tree presenting all the mergings performed) possible.

Agglomerative clustering firstly assigns each observation to its own cluster. In the next steps, pairs of clusters that are closest to each other are recursively merged into one cluster, until there is only one cluster left, containing all the points. Passing the `n_clusters` parameter is possible to obtain sensible division, however, it is not obligatory. Except for cluster assignment, there is an option to show all possible options to group data into a different number of clusters, by plotting a dendrogram presented in Figure 4.4.

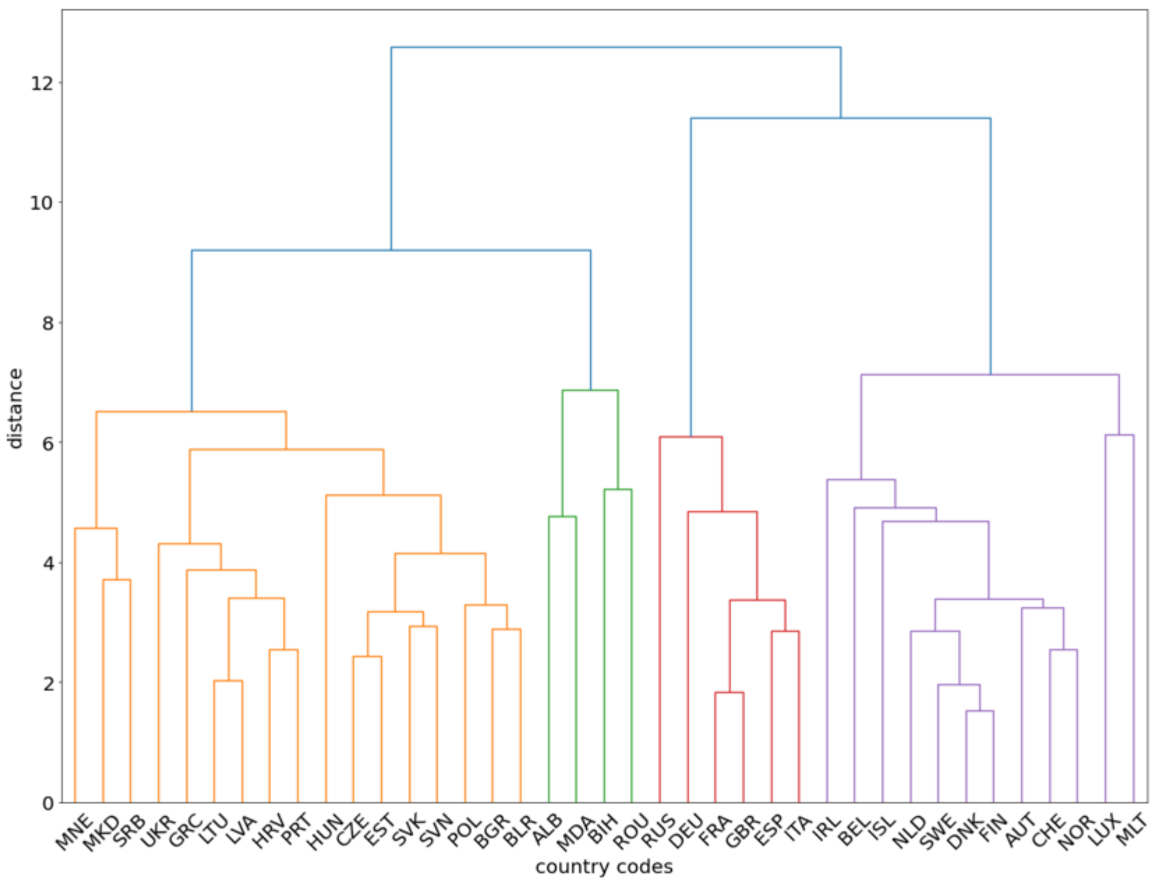


Figure 4.4: Exemplary dendrogram presenting results of agglomerative clustering algorithm

The optimal, yet again, a subjective number of clusters, based on dendrogram, is the one, where there is the largest difference in distance between merged clusters. Regarding Figure 4.4 there could be two candidates for the optimal number of clusters – three or four clusters. For a better comparison with K-means results, Figure 4.5 presents results for 4 clusters.

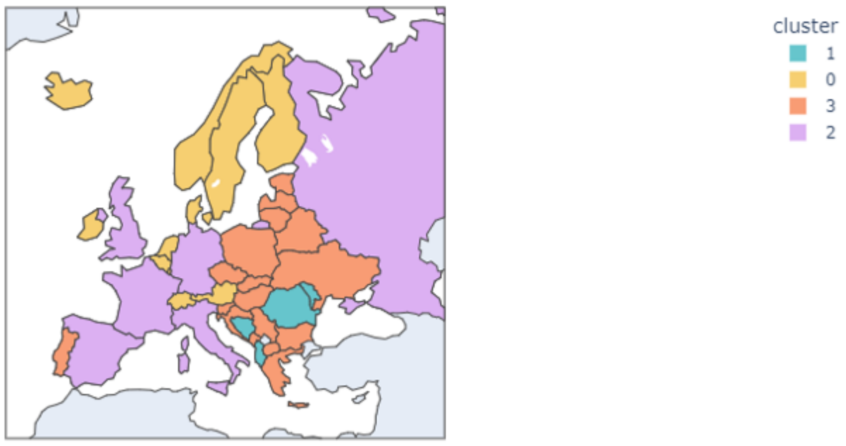


Figure 4.5: Results of Agglomerative clustering algorithm with 4 clusters specified and `linkage="complete"`

4.3. DBSCAN

DBSCAN is a density-based clustering algorithm, which is based on the idea of core samples – points in the areas of high density – and their neighbours; both groups being a part of a cluster. Proposed implementation uses the DBSCAN model from Python `scikit-learn` package and requires two parameters to be passed by the user:

- `eps` – the maximum distance between two points to consider one as a part of the other's neighbourhood; crucial to choose appropriately for the data and distance function
- `min_samples` – minimum number of points in the neighbourhood to consider the point as a part of the cluster; controls how tolerant the model is towards outliers

Higher `min_samples` value or lower `eps` value indicate higher density necessary to form a cluster. DBSCAN groups points from the same neighbourhood, which range and density are defined by parameters. The rest of the points that do not belong to any of the identified clusters are declared outliers. As well as in the case of the AgglomerativeClustering model, DBSCAN also does not support calculating distances for multivariate time series. Therefore, there is another important parameter: metric, permanently set to `precomputed`, which means that instead of data represented by data frame with observations and features, previously calculated distance matrix is passed as an argument. As mentioned in the previous chapter, the distance matrix is calculated using Python `dtaidistance` package, DTW algorithm. In the case of DBSCAN, it is theoretically impossible to force a preferred number of clusters. To find the desired results for this algorithm, some combinations of parameters need to be tested. During that process, the count of unique labels returned by the algorithm can be checked to find the best parameters provided that there are at least, for instance, 2 clusters formed.

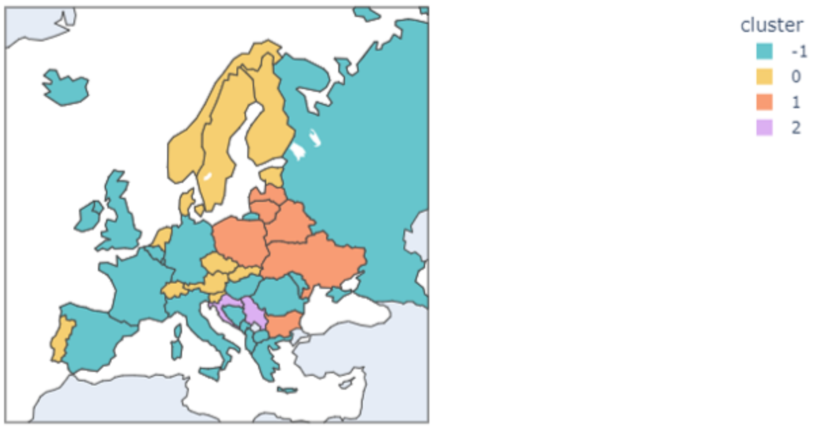


Figure 4.6: Results for DBSCAN for at least 2 clusters, `eps = 3.1` and `min_samples = 6`. The green colour denotes outliers

4.4. Hyperparameters tuning

The ranges of hyperparameters for tuning was set according to dataset characteristics. There are 39 countries, therefore in order to obtain sensible division, `n_clusters` parameter minimum value equals 2 and maximum value equals 8, and DBSCAN `min_samples` parameter values range from 2 to 10. Linkage parameter can be equal to any option possible in the `scikit-learn` AgglomerativeClustering model except for `ward` as explained in the model description. In the case of the DBSCAN `eps` parameter, its possible values are selected based on the distance matrix, in which the biggest distance between two counties equals approximately 12.

- KMeans:
  - n_clusters: iterate from 2 to 8, with step 1
- AgglomerativeClustering:
  - n_clusters: iterate from 2 to 8, with step 1
  - linkage: iterate through [‘complete’, ‘single’, ‘average’]
- DBSCAN:

- min_samples: iterate from 2 to 10, with step 1
- eps: iterate from 0.1 to 10, with step 0.1

Iterating through possible values of hyperparameters, for each one (or pair) clustering model is created, and Calinski-Harabasz score is measured for obtained cluster assignment. Based on its value, parameters that lead to the best clustering are chosen.

5. Analysis of the solution

5.1. Model evaluation

Clustering algorithms are evaluated using the existing cluster analysis assessment indexes. Only internal validation measures are used, which evaluate the goodness of a clustering structure without any additional (ground truth) information.

- **Silhouette score**. Higher value relates to a model returning clusters of better quality, and the value range is $[-1, 1]$. It is calculated by the following formula:

$$S = \frac{1}{n} \sum_i \frac{b_i - a_i}{\max(a_i, b_i)} \tag{5.1}$$

  where a_i stands for the average distance between points within the ith cluster, b_i stands for the average distance between ith cluster and other clusters, and the sum is over all clusters created.

- **Calinski-Harabasz index**. Good clustering with clearly separated clusters has a higher value of this index and the value range is $[0, \infty)$. Its formula is:

$$CH = \frac{\frac{SSB}{k-1}}{\frac{SSE}{k}} \tag{5.2}$$

  where SSB stands for the sum of squares between groups, SSE is the sum of squared errors (within the cluster), and k is the number of clusters.

- **Dunn index**. Its range is $[0, \infty)$ and it should be maximized. It is calculated with the formula:

$$D = \min_{1<i<k} \left\{ \min_{1<j<k, i \neq j} \left\{ \frac{\delta(C_i, C_j)}{\Delta C} \right\} \right\} \tag{5.3}$$

  where $\delta(C_i, C_j)$s for the smallest distance between data from different clusters and ΔC is the largest distance between clusters.

We start analysing results of the solution with comparing clustering algorithms' scores. To receive comparable results, all of the algorithms are performing clustering using parameters tuned according to silhouette score and data preprocessed with DTW similarity measure.

Table 5.1: Evaluation of different clustering algorithms.

algorithm	n_clusters	silhouette	CH score	Dunn index
KMeans	3	0.289	52.839	0.294
Hierarchical average	3	0.314	6.572	0.415
Hierarchical complete	4	0.311	62.274	0.380
Hierarchical single	2	0.380	4.298	0.511
DBSCAN	2	0.188	37.455	0.289

Notes: Table presents clustering results. Every algorithm performed clustering using parameters tuned according to silhouette score and distance was calculated using Dynamic Time Warping measure.

Table 5.1 presents the algorithms' results received using parameters tuned by silhouette score. All of the algorithms divide European countries into 2-4 groups, which agrees with the previously mentioned papers. Hierarchical single algorithm reached the best silhouette and Dunn index scores, however has the worst Calinski-Harabasz score. This might mean, that groups created by a Hierarchical single algorithm are well separated, but might contain less similar countries inside.

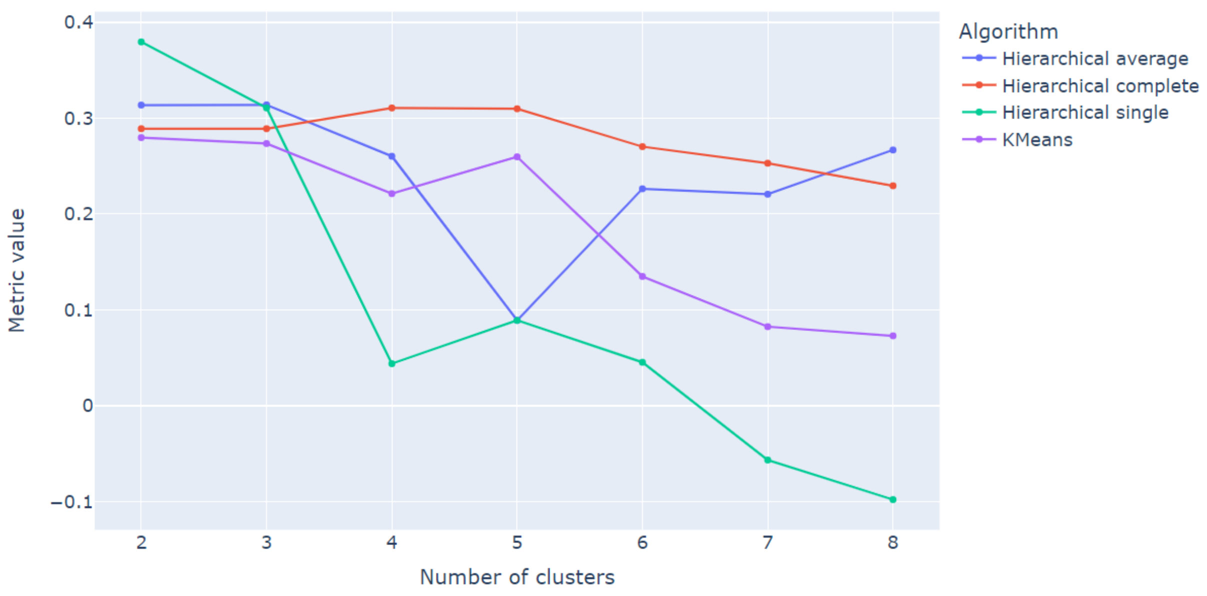


Figure 5.1: Silhouette scores for K-means and Hierarchical algorithms for different number of clusters selected.

Figure 5.1 presents KMeans and Hierarchical algorithms' results for a different number of chosen clusters measured by silhouette score. In this comparison, a Hierarchical single algorithm reveals suspicious behavior, achieving relatively good results only for 2 or 3 clusters chosen. On the contrary, Hierarchical complete results are satisfying regardless of the number of clusters parameter, which makes it a safer option.

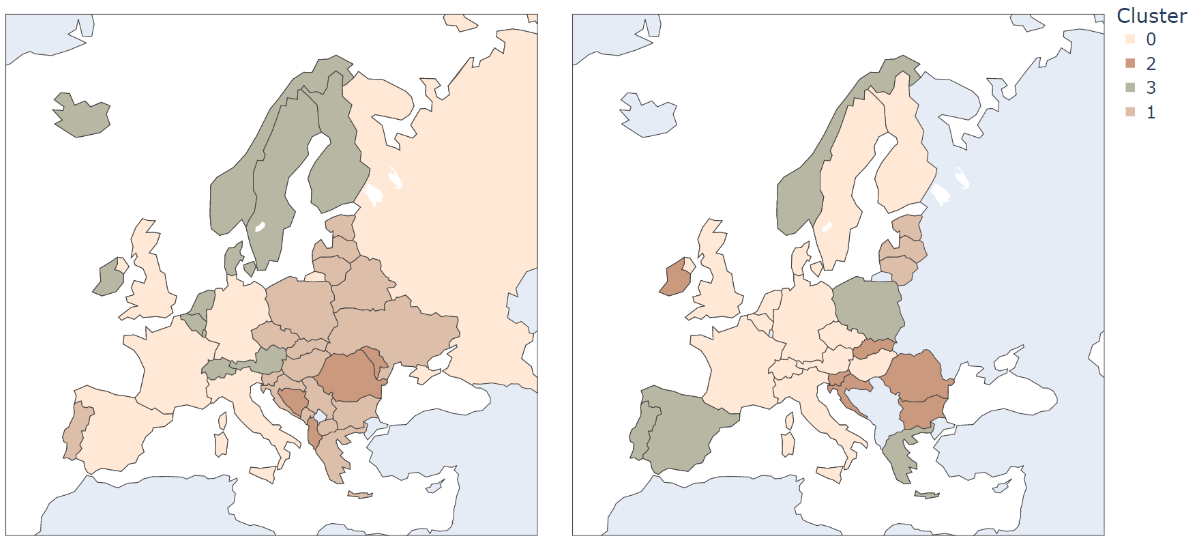


Figure 5.2: Comparison of clustering results. On the left, clustering results for hierarchical complete algorithm. On the right, division proposed in Markus Ahlborn and Marcus Wortmann's (2018) thesis [1]

This paragraph is dedicated to defining different clusters which are frequently indicated by our algorithms. Starting with defining the 'core' group of countries, most of the tested algorithms with different sets of parameters match Germany, France, Spain, Italy, and United Kingdom in one cluster. Apart from Russia, which is included in the same group by some of the algorithms e.g. Hierarchical complete, they are the biggest countries by population number in Europe and they may be considered as the 'core' group in our thesis. Another frequently created group by clustering algorithms includes Scandinavian countries, Austria, Switzerland, Belgium, and the Netherlands. These countries are less populated than members of the previously observed group, but they have bigger Human Development Index (HDI) scores and significantly larger Gross Domestic Product (GDP) per capita. This cluster may be called the 'developed' group as their economies seem to be on another level comparing to the rest of Europe. Eventually, the third big group is created by Central and Eastern Europe (CEE) countries. Members of this cluster were part of the Soviet Union's sphere of influence before 1990. The consequences may be observed up to this day as the GDP per capita and HDI scores are lower in this group.

Finally, the clustering algorithms' results need to be compared to the theory created in previous papers on this topic. In Figure 5.2 there are presented results of Hierarchical complete algorithm compared to the groups observed in Markus Ahlborn and Marcus Wortmann's (2018) thesis [1]. Their clustering results group countries, which business cycles synchronize. The most numerous group presented in their division consists of countries from 'core' and 'developed' groups defined in the previous paragraph. It appears these countries have another thing in common.

5.2. Results - comparison of various processing pipelines

This section describes results of the comparison of different approaches applied to the input data in terms of distance measurement and data transformations, as well as interesting insights gained during the project.

First aspect to analyze is the choice of the method of calculating distance between the countries. One of the approaches is the Euclidean distance - the most common metric used in various categories of problems. The second one is the Dynamic Time Warping - more adequate option for time series data, because distance between two sequences that may vary in time or speed (Chapter 4).

In order to compare these two methods, we conducted an experiment on the data resulting from the default preprocessing pipeline consisting of imputation, normalization and smoothing (Section 3.3). Distance between the countries was measured according to two mentioned approaches (Euclidean and DTW). For each option we performed modeling using all algorithms, with all the parameters from the grids described in Section 4.4. Then, the best clustering for each distance calculation method and each algorithm (algorithm version in case of hierarchical clustering) was chosen, based on the silhouette score. Table 5.2 presents the results of the experiment as well as the number of clusters for the selected groupings.

Table 5.2: Evaluation of different distance calculation approaches.

algorithm	n_clusters		silhouette		CH score		Dunn index	
	DTW	EUC	DTW	EUC	DTW	EUC	DTW	EUC
KMeans	3	6	0.289	0.320	52.839	42.725	0.294	0.394
Hierarchical average	3	3	0.314	0.314	6.572	6.571	0.415	0.415
Hierarchical complete	4	4	0.311	0.308	62.274	74.734	0.380	0.363
Hierarchical single	2	2	0.380	0.379	4.298	4.290	0.511	0.511
DBSCAN	2	2	0.188	0.189	37.455	37.439	0.289	0.289

Notes: Hierarchical clustering algorithm results are separated according to linkage criterion - hierarchical average denotes hierarchical clustering algorithm with `linkage` parameter set to `average` etc.

Comparing the metrics' values we can observe, that, according to silhouette score, there are no significant differences between the quality of groupings performed using different distance measures. In fact, in case of two versions of hierarchical clustering algorithm - with `linkage` parameter set to `average` and `single` - and DBSCAN algorithm, metric values are almost identical for both distance calculation methods. Moreover, this observation also applies to other metrics' values. As for the KMeans algorithm and hierarchical clustering algorithm with `linkage` set to `complete`, we can observe the biggest differences in case of Calinski-Harabasz score - for the first one usage of DTW leads to better results, whereas in the other, on the contrary. To sum up, it is difficult to explicitly determine which approach leads to better results based only on the metrics' values.

Second important aspect to analyze is the choice of preprocessing methods applied on the data in order to improve models' quality. To explore the influence of performed transformations on the clustering results, four pipelines were created, containing different preprocessing techniques described in detail in Section 3.3. All the pipelines include imputation as the first step. Afterwards, following approaches were implemented and analyzed:

- normalization and smoothing (default)
- normalization and logarithm transform
- normalization and Box-Cox transform
- outliers replacement and normalization

In order to compare these four approaches, we conducted an experiment on the data resulting from applying each of the mentioned preprocessing pipelines to the input data. As a distance calculation method, Dynamic Time Warping was selected. On each resulting dataset we performed modeling using all algorithms, with all the parameters from the grids described in Section 4.4. After that, the best clustering for each preprocessing technique and each algorithm (algorithm version in case of hierarchical clustering) was chosen, based on the silhouette score. Table 5.2 presents the results of the experiment as well as the number of clusters for the selected groupings.

Table 5.3: Evaluation of different data preprocessing approaches.

algorithm	data	n_clusters	silhouette	CH score	Dunn index
KMeans	default	3	0.289	52.839	0.294
	box-cox	2	0.091	48.269	0.176
	log	2	0.289	64.570	0.237
	outliers	3	0.289	52.839	0.294
Hierarchical average	default	3	0.314	6.572	0.415
	box-cox	2	0.523	23.191	0.196
	log	2	0.322	3.837	0.455
	outliers	3	0.310	6.421	0.423
Hierarchical complete	default	4	0.311	62.274	0.380
	box-cox	5	0.409	98.027	0.243
	log	4	0.313	75.468	0.386
	outliers	5	0.286	64.053	0.387
Hierarchical single	default	2	0.380	4.298	0.511
	box-cox	3	0.245	14.826	0.196
	log	2	0.322	3.837	0.455
	outliers	2	0.397	4.384	0.537
DBSCAN	default	2	0.188	37.455	0.289
	box-cox	2	0.377	31.993	0.113
	log	2	0.213	37.686	0.289
	outliers	2	0.179	36.665	0.294

Notes: Hierarchical clustering algorithm results are separated according to linkage criterion - hierarchical average denotes hierarchical clustering algorithm with `linkage` parameter set to `average` etc.

Comparing the number of clusters created by each approach, we can observe, that only for hierarchical clustering algorithm with `linkage` set to `complete` we obtain the best clustering with more than 3 clusters. Additionally, this method results in highest Calinski-Harabasz score values, no matter the preprocessing pipeline used. The best one, according to two metrics, is the clustering performed on the data after Box-Cox transformation. Similarly, this pipeline leads to best results also among clusterings created by hierarchical clustering algorithm with `linkage` set to `average`. In case of DBSCAN algorithm, for each preprocessing pipeline division into 2 clusters was selected as the best, with the grouping performed on the data after Box-Cox transformation being also the best but only according to silhouette score. Interestingly, for all algorithms groupings performed after default pipeline and the one with outliers replacement obtain almost identical results, which means their presence in the data probably neither influence negatively the quality of the clustering, nor conveys very important information leading to better distinction of the countries. Moreover, we also obtain similar results for logarithm transformation pipeline.

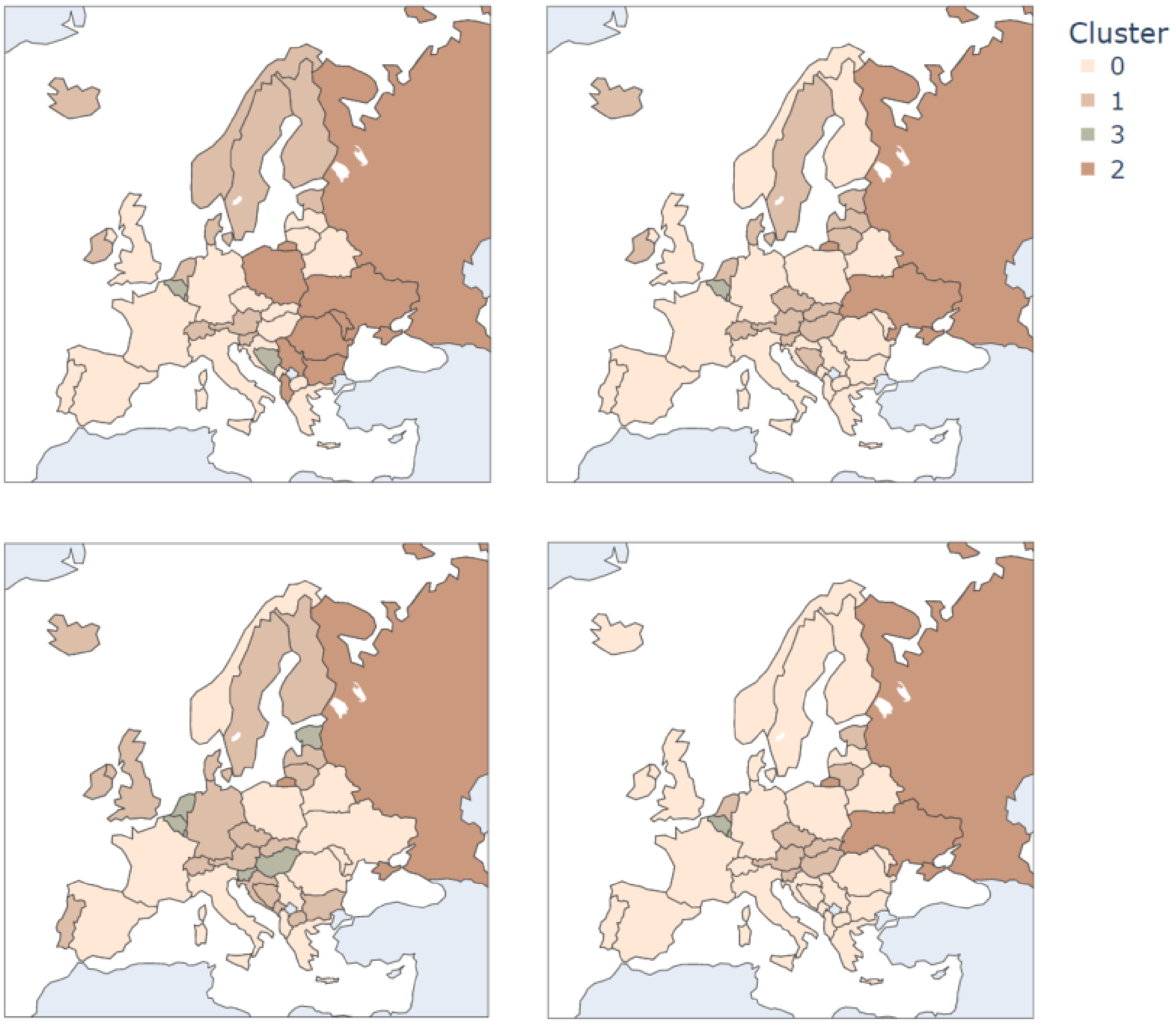


Figure 5.3: Hierarchical complete clustering results after box-cox transformation performed at 10-year intervals. Upper left image is for 1990-1998 period, upper right for 1997-2007 period, lower left for 2000-2010 period, and lower right for 2006-2016 period

At the end of the analysis of the solution, we want to present how clustering results depend on the period selected for the research. Figure 5.3 presents the grouping resulting from the Hierarchical complete clustering model after box-cox transformation. Grouping was performed at 10-year intervals, to observe the change happening in the economic indicators' values and their similarity between countries. As it can be observed, in 1998, a year before the founding of the eurozone, 'core' European countries, which accepted the new currency, are clustered together. It is explainable, given the fact that there's a list of requirements concerning the country's economy, to join the eurozone. There is also a cluster formed by the previous eastern block countries. Going forward to 2007, which was three years after many countries joined the EU and some of them also the eurozone, there is no longer a clear separation of those countries - they start to synchronize with more developed countries. In 2010, three years after Global Financial Crisis (GFC), we can observe an inconsistency in the core European countries group. This division may be determined by how countries managed to overcome the crisis. Finally, by 2016 core group looks more similar to the group created for the 1997-2007 period.

6. Conclusion and critical discussions

The choice of distance calculation method proved to have less impact on the clustering results than expected, based on the fact that DTW measure is more adequate for time series problems. Both Euclidean distance and DTW distance on average lead to really similar results in terms of values of clustering internal evaluation measures. Apart from different distance measures, various preprocessing techniques were also analyzed and compared. Among others, Box-Cox transform proved to be the most useful. The pipeline which included this transform, alonside with normalization, improved clustering results quality for almost every algorithm used (except for hierarchical clustering algorithm with `linkage` set to `single`.

Regardless of the multiple approaches used during the study, there are three main clusters that are created by most of the approaches:

- **core group** - big European countries like Germany, France, Spain, Italy, and United Kingdom that have important role in European Union and eurozone policy
- **developed group** - countries with strong economies like Scandinavian countries, Austria, Switzerland, Belgium, and the Netherlands
- **CEE** - countries that were part of the Soviet Union's sphere of influence before 1990 like Poland, Czech Republic, Slovakia, and Balkans

Such division partially reflect groupings proposed in previously published papers.

Bibliography

[1] Markus Ahlborn, Marcus Wortmann, The core–periphery pattern of European business cycles: A fuzzy clustering approach, *Journal of Macroeconomics*, 2017, 13–17.

[2] A. Belke, C. Dominic, D. Gros, Ruhr Economic Papers, *Ruhr-Universitat Bochum*, 2016.

[3] Bayoumi and Eichengreen, Shocking aspects of European monetary integration, *Cambridge University Press*, 1993, 193–240.

[4] Pentecôtee and Huchet-Bourdon, Revisting the core-periphery view of EMU, *Econ*, 2012, 2382–2391.

[5] Luís Aguiar-Conraria ,Manuel M.F. Martins, Maria Joana Soares, Convergence of the Economic Sentiment Cycles in the Eurozone: A Time-Frequency Analysis, *J. Common Mark. Stud. 51*, 2013, 377–398.

List of symbols and abbreviations

DTW	Dynamic Time Warping
i.e.	that is (*latin id est*)
GDP	gross domestic product
TFP	total factor productivity
PPP	purchasing power parity
CGDPo	output-based real gross domestic product per capita
PWT	Penn World Table

List of Figures

List of tables

List of appendices

Appendix 1. Deployment documentation

Instruction for proper environment configuration presented below applies to working on Windows operating system. Approximately 8-9GB free disk space is required.

1. Install Python 3.8.2: `https://www.python.org/downloads/release/python-382/` (newer releases of Python 3.8 should also work, however it was not tested)
   - During installation, check the box "Add Python 3.8 to path"
2. Install R: `https://cran.r-project.org/bin/windows/base` (newest version 4.1.2 or any other not older than 3.6 works)
   - Follow default installation. After completion, open R (`C:\ProgramFiles\R\R-<version>\bin\R.exe`) and run following command: dir.create(Sys.getenv("R_LIBS_USER"), recursive = TRUE). This action will create a user library for R located in `C:\Users\<user>\Documents\R\win-library\<version>` .
3. Open "Edit system environment variables", go to "Environment variables" and add new user variable.
   - Name: R_LIBS_USER
   - Value: `C:\Users\<user>\Documents\R\win-library\<version>` (path to R user library)
4. Install Git: `https://git-scm.com/download/win`. Follow default installation.
5. Install Microsoft Visual C++ Build Tools: `https://visualstudio.microsoft.com/pl/visual-cpp-build-tools`
   - Run the installer. Select only first component ("Desktop development with C++") and then proceed with the default selection of checkboxes on the right. There should be no more than 7GB of data to download. At the end of the installation it is required to restart the system.

6. Install Google Chrome or Microsoft Edge browser.
7. Create Python virtual environment. Open Command Line, enter the desired location and run following command: `python -m venv <environment_name>`. This will create a directory with separate Python executable, to which all required packages will be installed.

Appendix 2. Installation instruction

1. Clone GitHub repository. Open Git Bash, navigate to the desired location and run following command: `git clone https://github.com/amakarewicz/BEngThesis` OR Download GitHub repository. Go to `https://github.com/amakarewicz/BEngThesis` -> Code -> Download ZIP. Extract the contents into desired directory.

2. Run Command Line (best as an Administrator) and enter the directory with the repository (`BEngThesis` if it is cloned / `BEngThesis-main` if downloaded).

   - Example: `C:\Users\agama\Documents\BEngThesis`.

3. Run `install_requirements.sh` file, adding path to created previously Python virtual environment directory and path to R directory as command arguments.

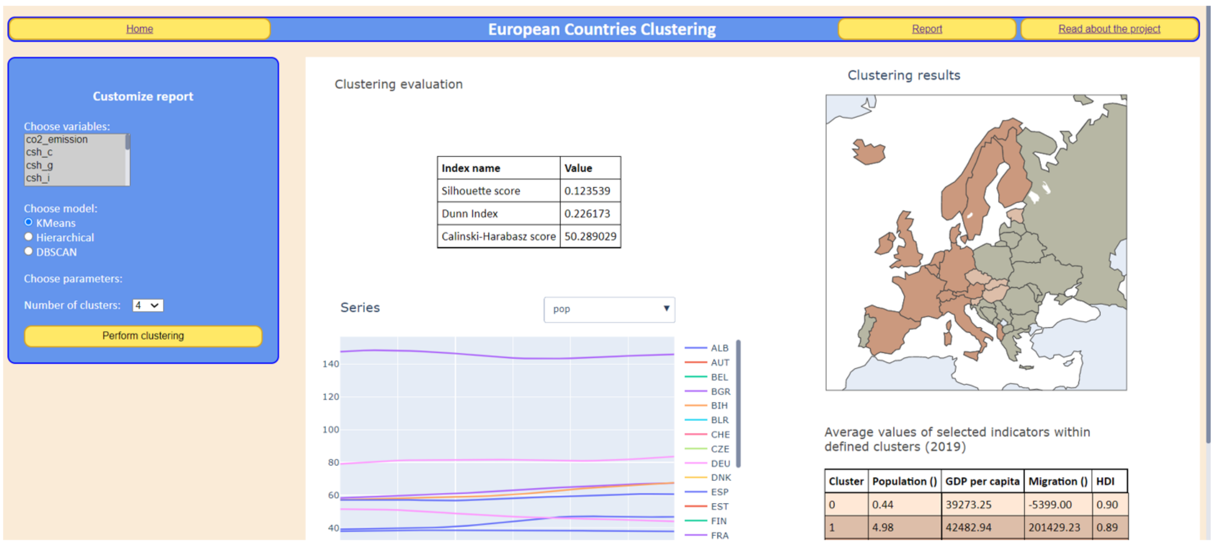
   - Command template: `install_requirements.sh "</path/to/python/env>" "</path/to/R>’’`
   - Example: `install_requirements.sh "C:\Users\agama\Documents\BEngThesis\django\bengthesis" "C:\ProgramFiles\R\R-4.0.0"`

4. Run start_app.sh file, adding path to created previously Python virtual environment directory as command argument (same as in the previous step).

   - Command template: `start_app.sh ‘‘</path/to/python_env>‘‘`
   - Example: `start_app.sh "C:\Users\agama\Documents\BEngThesis\django\bengthesis"`
   - If following warnings appear, do not worry, application is running.
     - UserWarning: h5py not installed, hdf5 features will not be supported.
     - UserWarning: R is not initialized by the main thread. Its taking over SIGINT cannot be reversed here, and as a consequence the embedded R cannot be interrupted with Ctrl-C. Consider (re)setting the signal handler of your choice from the main thread.

5. Run `http://127.0.0.1:8000/homepage` in your browser.

Appendix 3. User's manual



Clustering evaluation

Index name	Value
Silhouette score	0.123539
Dunn Index	0.226173
Calinski-Harabasz score	50.289029

Average values of selected indicators within defined clusters (2019)

Cluster	Population ()	GDP per capita	Migration ()	HDI
0	0.44	39273.25	-5399.00	0.90
1	4.98	42482.94	201429.23	0.89

To navigate between pages, use buttons in the top left and right corners of the page.

- To get information about the project, datasets and algorithms used, click "Read about project" button.
- To view evaluation of the created models and interesting results and insights, click "Report" button.
- To go back to home page, click "Home" button.

Home page

Use panel on the left to customize the report.

- To select multiple indicators, hold Ctrl button. All indicators are selected by default.
- Use radio buttons to select clustering algorithm. For each algorithm appropriate widgets are shown, enabling parameters choice. Use them to set parameters or leave the default ones.
- To generate results, click "Perform clustering" button.

To isolate only one country on the line chart, double click on the legend next to the trace assigned to it. Click on the trace in the legend to select/deselect a given country. Use a drop-down list to select indicator. Hover over the trace to see more information. Use the cursor to select and zoom desired part of the chart. Hover over the country on the map to see more information.

Report page

Use a drop-down list to select a metric according to which models were evaluated. Use rest of the functionalities of the line chart described above. Use a slider to manipulate parameters values / select desired time range. Hover over the country on the maps to see more information.

Appendix 4. GUI design

TODO