Faculty of Mathematics and Information Science

WARSAW UNIVERSITY OF TECHNOLOGY

Group Project
Documentation: part 3

Authors: *Agata Makarewicz, Jacek Wiśniewski*

Thesis title: *Application for Analysis of the Economic Growth Indexes for European Countries*

Supervisor: *Agnieszka Jastrzębska, Ph.D. Eng.*

version 1.3

24.11.2021

Table of Contents

1 Abstract

This document contains model descriptions for the engineering group diploma thesis entitled “Application for Analysis of the Economic Growth Indexes for European Countries”. It is a continuation of the previous document „Group Project Documentation: part 2”. The document is dedicated to a module containing clustering models, providing their descriptions, required parameters and exemplary results, as well as baseline evaluation and comparison. Furthermore, the application template and code with models implementation are an appendix to this part of the documentation.

1.1 History of changes

Date	Author	Description	Version
14.11.2021	Agata Makarewicz	Template	1.0
20.11.2021	Jacek Wiśniewski	Model descriptions	1.1
22.11.2021	Agata Makarewicz Jacek Wiśniewski	Maps, vocabulary update & final adjustments	1.2
24.11.2021	Agata Makarewicz, Jacek Wiśniewski	Hyperparameters tuning chapter added, evaluation chapter extended	1.3

2 Vocabulary

Homepage - a webpage presented after turning on the application. It will have all of the functionalities like filtering data and generating the report.

“Read about the project" page – a webpage that will present all of the information about the project, authors and contact email addresses.

Report – content from homepage consisting of charts and results of clustering algorithms with comments.

Clustering - the task of dividing a set of objects into several groups called clusters in such a way that objects within the same cluster are more similar to each other than to objects in other clusters.

Model – machine learning algorithm used for clustering.

Dendrogram – a diagram that shows the hierarchical relationship between objects. It is most commonly created as an output from hierarchical clustering.

Distance matrix - square matrix or a two-dimensional array containing the pairwise distances between the elements of a set

3 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 from described below.

- **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.
- **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.
- **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, it is essential to change it to a more adequate Dynamic Time Warping (DTW) metric. 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 1.

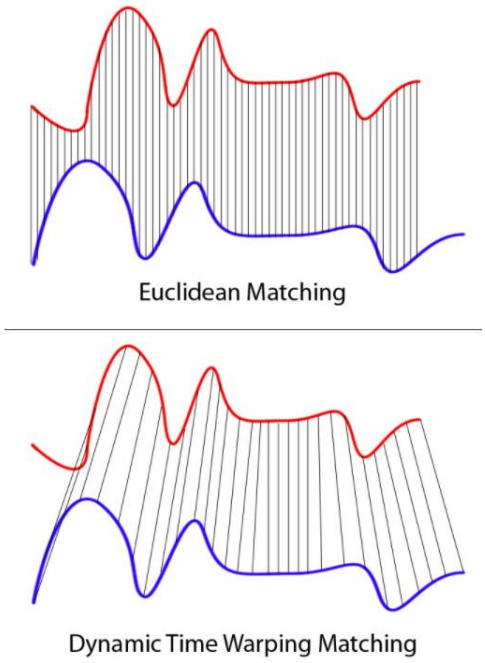

Figure 1. Difference between Euclidean Matching and DTW Matching

3.1 K-Means

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:

$$\sum_{i=0}^{n} \min_{\mu_j \epsilon \, C} (|| \, x_i - \mu_j ||^2)$$

where C - 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 2 can be used.

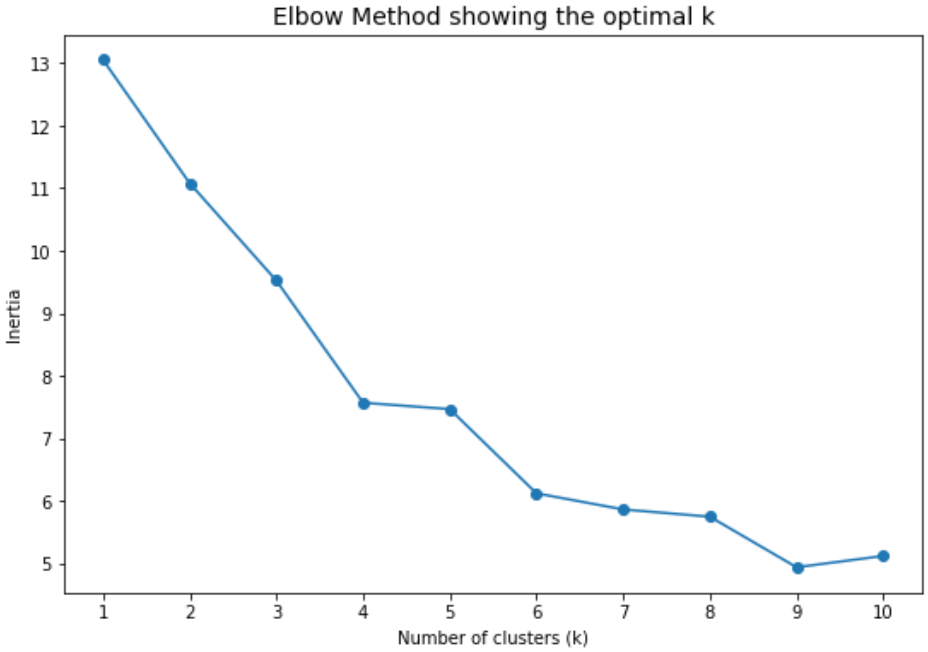

Figure 2. Inertia for different n_clusters parameter values, for the KMeans clustering, performed on the analyzed dataset.

Figure 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 2 where inertia starts to decrease slower, for instance, 4.

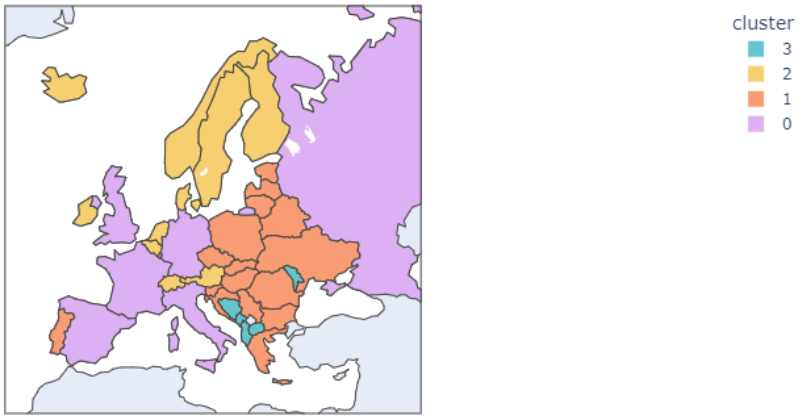

Figure 3. Results of K-means clustering for k=4

3.2 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.

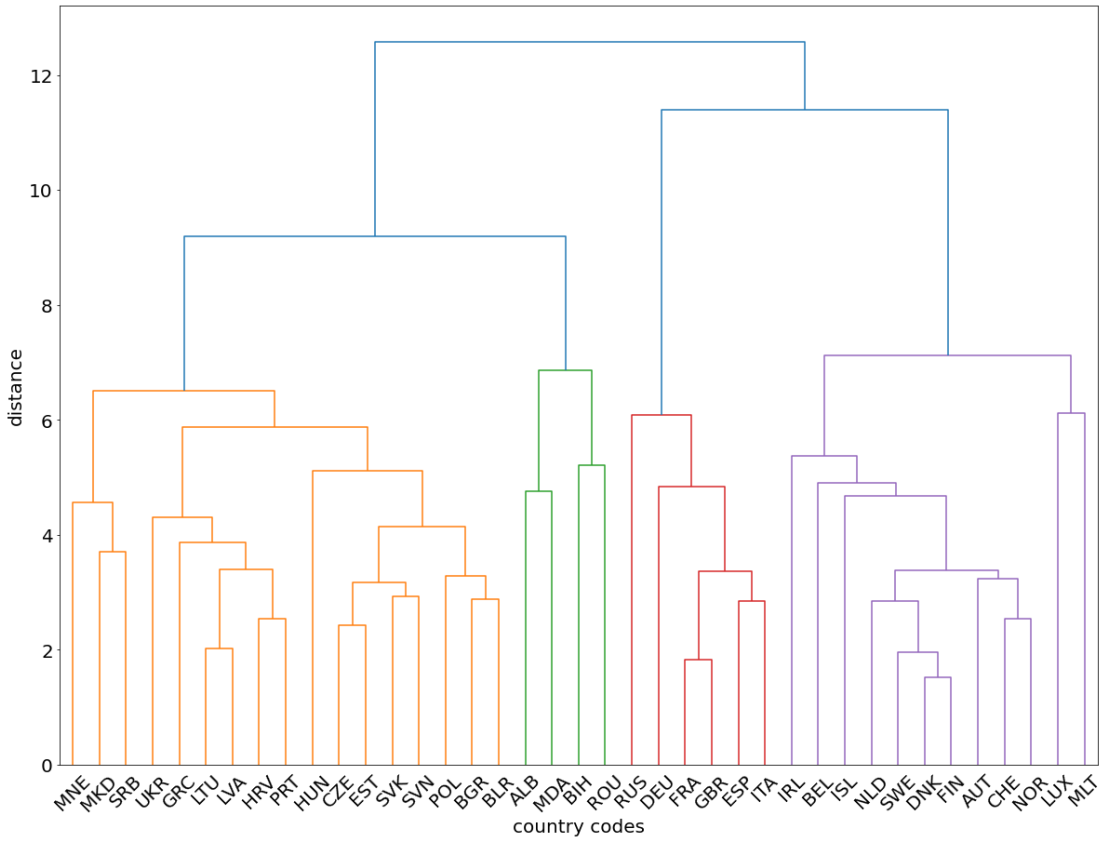

Figure 4. Exemplary dendrogram presenting results of agglomerative clustering algorithm

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 there could be two candidates for the optimal number of clusters – three or four clusters. For a better comparison with K-means results, Figure 5 presents results for 4 clusters.

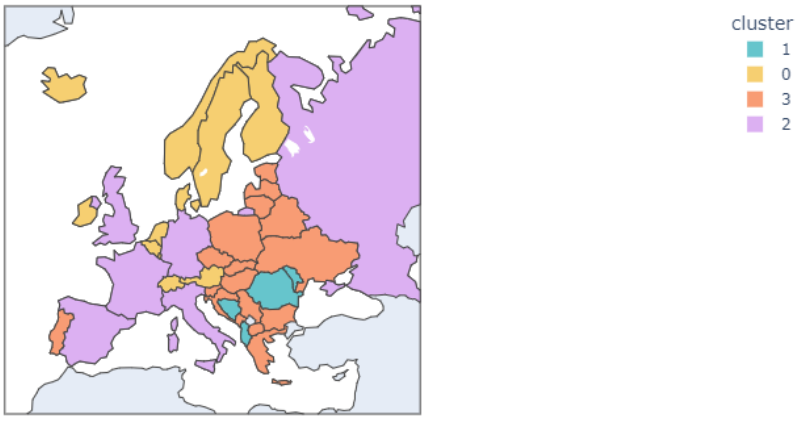

Figure 5. Results of Agglomerative clustering algorithm with 4 clusters specified and linkage='complete'.

3.3 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.

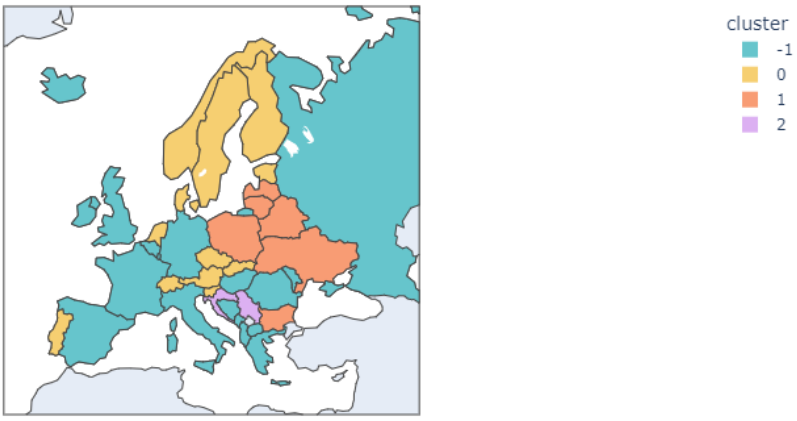

Figure 6. Results for DBSCAN for at least 2 clusters, eps = 3.1 and min_samples = 6. The green colour denotes outliers.

3.4 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 10, and DBSCAN *min_samples* parameter values range from 2 to 19. *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 10, with step 1
- AgglomerativeClustering:
  - *n_clusters*: iterate from 2 to 10, with step 1
  - *linkage*: iterate through ['complete', 'single', 'average']
- DBSCAN:
  - *min_samples*: iterate from 2 to 19, 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.

3.5 Baseline evaluation

Clustering algorithms will be 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:

$$S = \frac{1}{n}\sum_{i}\frac{b_i - a_i}{max\ (a_i, b_i)}$$

  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, ∞). Its formula is:

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

  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, ∞) and it should be maximized. It is calculated with the formula:

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

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

	KMeans (k=3)	*Agglomerative (k=4, linkage='complete')*	*DBSCAN (eps=3.1, min_samples=6)*
Silhouette score	0.103466	0.238829	-0.076521
Dunn index	0.364566	0.530005	0.593828
C-H score	58.823557	96.080558	59.134446

Table 1. Silhouette score, Dunn index and Calinski-Harabasz score values for K-means, AgglomerativeClustering and DBSCAN, for the clustering performed with optimal parameters selected during the tuning phase.

4 Bibliography

[1] Aghabozorgi, Saeed, Shirkhorshidi, Ali S., and Wah, Teh Y. Time-series clustering – A decade review. Information Systems 53 16-38, 2015.

[2] Gräbner, C., Heimberger, P., Kapeller, J., and Schütz B. Structural change in times of increasing openness: assessing path dependency in European economic integration. Journal of Evolutionary Economics 30, 1467–1495, 2020.

[3] Bartlett, W. and Prica, I. Interdependence between Core and Peripheries of the European Economy: Secular Stagnation and Growth in the Western Balkans. LSE'Europe in Question' Discussion Paper Series, LEQS Paper No. 104/2016, 2016.

[4] Hamilton, James Douglas Time Series Analysis. Princeton University Press, 1994.

[5] Pal, Avishek, Prakash, PKS. Practical Time Series Analysis. Master Time Series Data Processing Visualization and Modelling Using Python. Packt, 2017

[6] Fu-Lai Chung, Tak-Chung Fu, V. Ng and R. W. P. Luk, "An evolutionary approach to pattern-based time series segmentation," in *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 5, pp. 471-489, Oct. 2004.

[7] Scikit-learn developers (BSD License), Scikit-learn User Guide: 2.3 Clustering, 2007-2021, https://scikit-learn.org/stable/modules/clustering.html

[8] Jeremy Zhang, Dynamic Time Warping. Explanation and code. 2020, Feb 1 https://towardsdatascience.com/dynamic-time-warping-3933f25fcdd

[9] Xiaoji Wan, Hailin Li, Liping Zhang, Yenchun Jim Wu, "Multivariate Time Series Data Clustering Method Based on Dynamic Time Warping and Affinity Propagation", Wireless Communications and Mobile Computing, vol. 2021, Article ID 9915315, 8 pages, 2021. https://doi.org/10.1155/2021/9915315

[10] Siebert, J.; Groß, J.; Schroth,C. A Systematic Review of Python Packages for Time Series Analysis.Eng. Proc. 2021, 5, 22. https://doi.org/10.3390/engproc2021005022

[11] Julio-Omar Palacio-Nino, Fernando Berzal Evaluation Metrics for Unsupervised Learning Algorithms