POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

A Framework for Time-Series Clustering-as-a-service

Tesi di Laurea Magistrale in
Automation and Control Engineering - Ingegneria dell'Automazione

Author: **Uriel Guadarrama Ramirez**

Student ID: 943516
Advisor: Prof. Manuel Roveri
Academic Year: 2021-22

Abstract

Time-series clustering has effectively provided useful information in different application domains. There is an increased interest in time-series clustering as part of the effort in temporal data mining research. Clustering-as-a-service is an innovative and promising research area aiming at designing Cloud-based platforms and systems able to provide users an automatic tool to perform with high efficiency and quality time-series clustering in an "as-a-service" manner. This thesis introduces TIMEX-CLUSTERING, an open-source Python-based framework for time-series clustering-as-a-service. Differently from the tools and libraries present in the literature, which provide excellent results but based on non-automatic approaches, TIMEX-CLUSTERING provides, in an "as-a-service" manner, a fully automatic end-to-end clustering pipeline comprising data ingestion, data preprocessing, data description, data clustering and service delivery. In addition, w.r.t. the literature where only one clustering approach is chosen based on the subject data, TIMEX-CLUSTERING allows to use three different clustering approaches with a fully automatic pipeline, therefore the user can cover datasets with different nature and compare clusters obtained through different approaches. Finally, several experiments are performed and TIMEX-CLUSTERING is successfully applied to the analysis and clustering of the COVID-19 pandemic around the world.

Keywords: time-series, clustering, Cloud-computing, as-a-service

Abstract in lingua italiana

Il clustering di serie temporali ha effettivamente fornito informazioni utili in diversi domini applicativi. C'è un crescente interesse per il raggruppamento di serie temporali come parte dello sforzo nella ricerca di data mining temporale. Clustering-as-a-service è un'area di ricerca innovativa e promettente che mira a progettare piattaforme e sistemi basati su cloud in grado di fornire agli utenti uno strumento automatico per eseguire con alta efficienza e qualità il clustering di serie temporali in modalità "as-a-service". Questa tesi introduce TIMEX-CLUSTERING, un framework open source basato su Python per il clustering-as-a-service di serie temporali. A differenza degli strumenti e delle librerie presenti in letteratura, che forniscono ottimi risultati ma basati su approcci non automatici, TIMEX-CLUSTERING fornisce, in modalità "as-a-service", una pipeline di clustering end-to-end completamente automatica comprendente acquisizione dei dati, preelaborazione dei dati, descrizione dei dati, clustering dei dati e fornitura di servizi. Inoltre, rispetto alla letteratura in cui viene scelto un solo approccio di clustering in base ai dati del soggetto, TIMEX-CLUSTERING consente di utilizzare tre diversi approcci di clustering con una pipeline completamente automatica, quindi l'utente può coprire set di dati di diversa natura e confrontare i cluster ottenuti attraverso approcci differenti. Infine, vengono eseguiti diversi esperimenti e TIMEX-CLUSTERING viene applicato con successo all'analisi e al raggruppamento della pandemia di COVID-19 in tutto il mondo.

Parole chiave: time-series, clustering, Cloud-computing, as-a-service

Contents

1 | Introduction

Nowadays the sensors and tracking mechanisms are everywhere, and thanks to the increased power of data storage and processors, it has been possible to store and keep data for long periods of time; as a result, there are unprecedented amounts of high-quality data available, going from weather, stock markets, customer purchase histories to recordings of heartbeats; they all form time-series. Time-series data is everywhere, and this kind of data grows increasingly ubiquitous and important as the big data ecosystem expands. Time-series data spans a wide range of disciplines and use cases as mentioned. Any data that has an ordered axis can be analyzed with time-series methods, even if that ordered axis is not time per se.

Time-series are uniquely interesting, consequently many projects and research relevant to analyzing time-series have been performed in many areas with different purposes, such as: sub-sequence matching, anomaly detection, classification, visualization, segmentation, identifying patterns, trend analysis, summarizing, forecasting and clustering.

In recent years with the growth of cloud computing and big data and their vast applications, research interest increased on unsupervised solutions like clustering algorithms to extract knowledge from this huge amount of data. By using clustering time-series data patterns can be discovered (such as periodicity, seasonality, or cycles), which empower data analysts to extract valuable information from complex and massive datasets.

Clustering is a category of unsupervised learning techniques that allows us to discover hidden patterns in data where we do not know the right answer upfront. The goal of clustering is to identify homogeneous groups in data called *clusters*, so that objects in the same cluster have maximum similarity with other objects within the group, and minimum similarity with objects in other groups. This idea holds just as true for time-series data as for other kinds of data.

The analysis and research of the time-series clustering field is of great importance, because, as mentioned, it helps to obtain (through pattern discovery) valuable information. Moreover, it makes possible to handle very large databases, and it is the most used approach as an exploratory technique and as a subroutine in complex data mining algorithms, given

that it can help to users to quickly understand the structure of data, clusters, anomalies, and other regularities in datasets by representing time-series cluster structures as visual images.

1.1. Current technologies

The literature about algorithms and models for time-series clustering is wide and deep with relevant solutions being state-of-the-art in many application scenarios such as social sciences, medicine, biology, business, economics, astronomy, statistics, image recognition, processing of digital information, marketing, physics and many others.

In the recent decade, there has been a considerable number of changes and developments in the time-series clustering area with the goal to improve in efficiency, quality and complexity the clustering time-series approaches; this transformation was caused by emerging concepts such as big data and cloud computing which increased size of data sets exponentially. There are generally three different ways to cluster time-series, namely shape-based or observation-based, feature-based and model-based, as well as their enhancement methods, which will be explained with detail in the *Chapter 2*.

The observation-based approach is a direct approach for clustering time-series based on a comparison of the observed time-series or a suitable transformation of the observed time-series [49]. The observation-based approach is recommended when the aim is to identify similar geometric profiles and when the time-series are not very long.

The feature-based metrics are more suitable when the aim is to discriminate between generating processes, that is, between underlying dependence structures [49]. In this approach the clustering target is achieved by using the features extracted, these features highlight higher level dynamic aspects of the time-series; through this approach a dimensional reduction is attained and consequently computational time saving can be achieved.

Finally, in the model-based methods, the time-series under consideration are assumed to have been generated from specific underlying models or by a combination of probability distributions, and the similarity between fitted models is evaluated [49].

1.2. Approach of this thesis

The growth in interest in time-series clustering has resulted in the development of a wide variety of techniques. However, as shown in the vast volume of time-series literature, traditional time-series analysis and modeling tend to be based on non-automatic and trial-and-error approaches; therefore, to achieve an efficient and high-quality time-series clustering, the first challenge is to choose which algorithm to use among all the broad spectrum of techniques. The choice of clustering algorithm depends both on the type of data available and on the particular purpose and application. The second challenge is to accomplish the clustering in a fully automatic end-to-end pipeline comprising the different steps involved (such as data ingestion, data preprocessing, data description, data clustering and service delivery) according on the clustering approach chosen.

The purpose of this thesis is to develop a new clustering platform able to provide users an automatic tool to perform time-series clustering with high efficiency and quality. This framework could be implemented for any user with a minimal amount of knowledge about clustering procedures, and it will provide the template sets as accurate as those created by other clustering algorithms. An analogous innovation but in the field of time-series forecasting, called TIMEX [2], provided in an "as-a service" manner a fully automatic end-to-end forecasting pipeline.

This thesis introduces TIMEX-CLUSTERING, an open-source Python-based framework for time-series clustering-as-a-service. Differently from tools and libraries present in the literature, which provide excellent results only for specific cases, TIMEX-CLUSTERING provides, in an "as-a service" manner, a fully automatic end-to-end clustering pipeline comprising all the steps involved to cluster time-series (such as data ingestion, data preprocessing, data description, data clustering and service delivery).

TIMEX-CLUSTERING uses the three different clustering approaches (observation-based, feature-based and model-based), in order to cover datasets with different nature. The proposed TIMEX-CLUSTERING framework has been built with the goal of providing an open-source framework able to offer in a fully automatic pipeline all the relevant characteristics that an as-a-service framework should have.

1.3. Outline of the thesis

The thesis is structured as follows:

- *Chapter 2* presents the state of art of time-series clustering, introducing the concepts to understand the unsupervised clustering techniques, and it explains the observation-based, feature based and model-based clustering approaches.
- *Chapter 3* presents the proposed Timex-Clustering as-a-service framework, explaining every detail and foundation of the framework.
- *Chapter 4* explains the details of the datasets that were used and the criteria by which these datasets were chosen.
- *Chapter 5* describes the assessments that were performed using TIMEX-CLUSTERING and shows the results obtained.
- *Chapter 6* summarizes the entire work, drawing conclusions on our approach to the problem.

2 | Related literature

In this chapter a survey of the three main approaches that have been used in literature for time-series clustering is presented, as well as the different techniques that compose them. These algorithms and models have been successfully applied in many clustering application scenarios, such as social economics, medicine, biology, business, management, statistics, image recognition, marketing, physics and many others.

2.1. Time-Series Clustering

A *time-series* is a sequence $\{y_t\}$ of values assumed by a quantity of interest that can be measured at specific time periods t. If the time indices t for which the measurements y_t are gathered belong to a discrete set T, the time-series $\{y_t\}$ is called a *discrete time-series*; otherwise it is called a *continuous time-series* [52]. In what follows, we will be working with mainly discrete time-series which are realizations of discrete stochastic processes, reminding that a discrete stochastic process is defined as a collection of random variables that are ordered in time and defined as a set of points which are discrete.

Given a data set of n time-series data, $D = \{F_1, F_2, \ldots, F_n\}$, the process of unsupervised partitioning of D into $C = \{C_1, C_2, \ldots, C_k\}$, in such a way that homogeneous time-series are grouped together based on a certain similarity measure, is called *time-series clustering*. The goal of clustering is to identify homogeneous groups in data called *clusters*, so that objects in the same cluster have maximum similarity with other objects within the group, and minimum similarity with objects in other groups.

Clustering time-series data has become a popular research topic over the past decades ([1],[8],[37],[49],[52]) and there is a rich literature on this topic. Clustering is a category of unsupervised learning techniques that allows us to discover hidden patterns in data; these patterns can commonly take place in the dataset. Finding the clusters of time-series can help to real world problems such as anomaly detection (i.e. discover anomalies in sensor databases), recognizing dynamic changes in time-series (i.e. in financial data bases to find companies with similar stock price move), forecasting (i.e. infer likely future behavior

based on cluster membership) and for pattern discovery (i.e. in marketing databases different daily patterns of sales can be discovered).

The clustering of time-series is organized into three approaches depending upon whether they work directly on raw data, indirectly with the features extracted from the raw data or with model built from raw data; the methods used can work either in frequency, time or wavelet domain. These approaches are named shape-based or observation-based, feature-based and model-based respectively; the Figure 2.1 extracted from [1] shows a brief scheme of these approaches. The major issues related to time-series clustering are high dimensionality, temporal order and noise, which can significantly affect the quality of clusters formed.

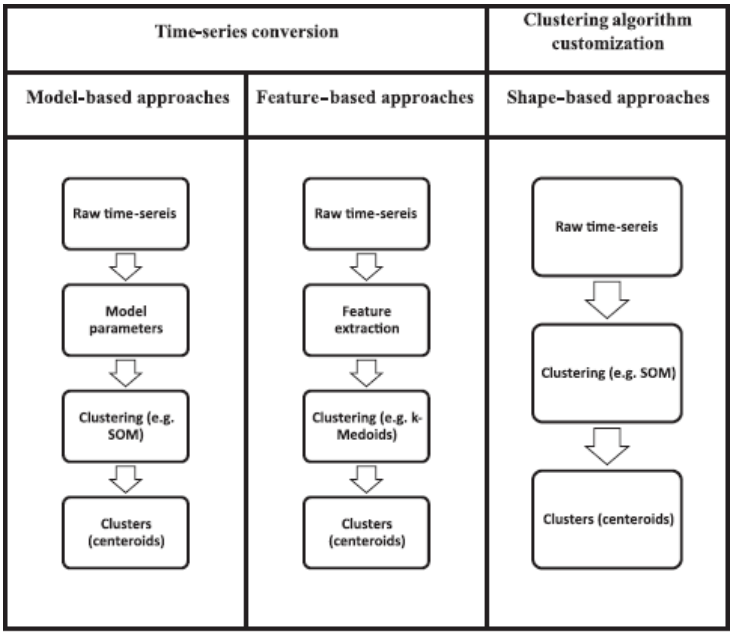


Figure 2.1: The time-series clustering approaches [1].

There exist two key aspects to achieve an effective and efficient time-series clustering, which are time-series representation methods and similarity measures. Moreover, there are some criteria to evaluate the performance of time-series clustering methods, which will be explained also in this chapter.

2.1.1. Representation methods

Time-series are high dimensional data and dealing directly with such data in its raw format is very expensive computationally. Besides, when measuring the distance of raw time-series, some distance measures are highly sensitive to noise in the data, and consequently the time-series that are similar in noise could be being clustered instead of clustering them based on similarity in shape. Therefore, representation techniques that reduce the dimensionality of time-series have been developed, while still preserving the fundamental characteristics of a particular data set. Dimensionality reduction represents the raw time-series in another space by transforming time-series to a lower dimensional space.

Definition: Given a time-series data $F_i = \{f_1, f_t, \ldots, f_T\}$, a representation technique allows to transform the time-series to another dimensionality reduced vector $F_i = \{f_1, \ldots, f_x\}$ where $x < T$ and if two series are similar in the original space, then their representations should be similar in the transformation space too [1].

The most common techniques are: Discrete Fourier Transformation (DFT), Single Value Decomposition (SVD), Discrete Cosine Transformation (DCT), Discrete Wavelet Transformation (DWT), Piecewise Aggregate Approximation (PAA), Chebyshev polynomials (CHEB), Indexable Piecewise Linear Approximation (IPLA), Symbolic Aggregate Approximation (SAX), Adaptive Piecewise Constant Approximation (APCA), Piecewise Linear Approximation (PLA), etc.

Each time-series dimensionality reduction technique has its own features and drawbacks; some describe the dynamics of a stationary time-series in the time domain, whereas others (such as DFT) describe in the frequency domain the dynamics of a stationary time-series, also DWT can be used for stationary and non-stationary time-series. The techniques (DFT, SVD, DCT, DWT, PAA, CHEB) are suitable for time-series with fixed size (equal-length) segmentation, which make the comparison of representations of several time-series straightforward; while other techniques (PLA, APCA, SAX) can better approximate each series but the comparison of several time-series is more difficult. As mentioned before, choosing an appropriate data representation method is a key component which affects the efficiency and accuracy of the solution, in *Chapter 3* the techniques chosen and the corresponding reasons to do it will be explained in detail.

2.1.2. Distance/Similarity measures

There are different measures which can be applied to measure the distance among time-series. Some similarity measures are proposed based on specific time-series representations

and some others work regardless of representation methods. One of the simplest ways for calculating distance between two time-series is considering them as univariate time-series, and then calculating the distance measurement across all time points.

Definition: A univariate time-series is a sequence of real numbers collected regularly in time, where each number represents a value [51].

Definition: Let $F_i = \{f_{i1}, f_{it}, \ldots, f_{iT}\}$ and $F_j = \{f_{j1}, f_{jt}, \ldots, f_{jT}\}$ be time-series of length T. If the distance between two time-series is defined across all time points, then $dist(F_i, F_j)$ is the sum of the distance between individual points [1].

$$dist(F_i, F_j) = \sum_{t=1}^{T} dist(f_{it}, f_{jt})$$

There are several similarity/distance measures for similarity of time-series data already proposed, the most common are: Euclidean distance (ED), Dynamic Time Warping (DTW), distance based on Longest Common Subsequence (LCSS), Edit Distance with Real Penalty (ERP), Edit Distance on Real sequence (EDR), DISSIM, Pearson's correlation coefficient and related distances, Hausdorff distance and modified Hausdorff (MODH).

Measuring the distance of time-series involves facing challenges such as noise, amplitude scaling, offset translation, longitudinal scaling, linear drift, discontinuities and temporal drift which are the common properties of time-series data. The choice of a proper distance approach depends on the characteristic of the time-series, length, representation method and on the objective of the clustering.

The objective of the clustering could be of three types. The first type consists in finding similar time-series in time; correlation base or ED distance are proper for this objective. This kind of distance is costly on raw time-series, so the calculation is performed on transformed time-series, such as DFT, DWT or PAA. The second type of objective consists in finding similar time-series in shape; the time of occurrence of patterns is not important. As a result, methods such as Dynamic time Warping (DTW), LCSS are used for dissimilarity calculation. The third objective consists in finding similar time-series in change (structural similarity); modelling methods such as Hidden Markov Models (HMM) or an ARMA process are utilized, and then similarity is measured on the parameters of fitted models.

2.1.3. Clustering methods

Han and Kamber [25] classified clustering methods developed for handing various static data into five major categories: traditional methods, partitioning methods, density-based methods, grid-based methods, and model-based methods. Three of the five major categories of clustering methods for static data, specifically traditional methods (hierarchical methods), partitioning methods (non-hierarchical) and model-based methods, have been utilized directly or modified for time-series clustering [37].

Traditional clustering (Hierarchical methods)

The traditional cluster analysis, also called hierarchical clustering, is used for clustering time-series. In this case, first a suitable distance measure inheriting the dynamic features of the time-series is defined for comparing time-series and, subsequently, a standard hierarchical cluster analysis is applied using the defined distance.

Hierarchical clustering groups data with a sequence of nested partitions, either from singleton clusters to a cluster including all individuals (agglomerative clustering) or vice versa (divisive clustering). Both agglomerative and divisive clustering methods organize data into the hierarchical structure based on suitable proximity measures. Agglomerative method is probably the most widely used of the hierarchical methods, it considers each item as a cluster, and then gradually merges the clusters (bottom-up). The merging of a pair of clusters or the formation of a new cluster is dependent on the definition of the distance function between two clusters. The most common methods for distance function are:

- Single linkage method: the distance between a pair of clusters is determined by the two closest units to the different clusters. If the clusters are separated far from each other, the single linkage method works well; this is not the case if the clusters are too close and there is noise [49].
- Complete linkage method: it uses the farthest distance of a pair of objects to define inter-cluster distance [49].
- Group average linkage method: the distance between two clusters is defined as the average of the distances between all pairs of data points [49].
- Weighted average linkage method: the average linkage is also used to calculate the distance between two clusters. The difference is that the distances between the newly formed cluster and the rest are weighted based on the number of data points in each cluster [49].

- Centroid linkage method: two clusters are merged based on the distance of their centroids (means) [49].
- Ward's method (minimum variance method): the aim of Ward's method is to minimize the increase of the so-called within-class sum of the squared errors [49].

Single linkage, complete linkage, and average linkage consider all points of a pair of clusters when calculating their inter-cluster distance, and they are also called graph methods. The others are called geometric methods because they use geometric centers to represent clusters and determine their distances.

For hierarchical clustering methods, the optimal partition is achieved by selecting one of the solutions in the nested sequence of clusterings that the hierarchy comprises. Large changes in fusion levels are taken to indicate the best cut.

The hierarchical algorithms are weak in terms of quality because they cannot adjust the clusters after splitting a cluster in divisive method, or after merging in agglomerative method. The results of hierarchical clustering are usually depicted by a particular graphic called dendrogram, this representation provides very informative descriptions and a visualization of the potential data clustering structures, especially when real hierarchical relations exist in the data. Additionally, in contrast to most algorithms, hierarchy clustering does not require the number of clusters as an initial parameter which is an outstanding feature of this algorithm, because usually it is hard to define the number of clusters in real world problems. Moreover, despite many algorithms, hierarchical clustering has the ability to cluster time-series with unequal length if an appropriate elastic distance measure such as DTW or LCSS is used to compute the similarity of the time-series. The fact that prototypes are not necessary in its process has made this algorithm capable to accept unequal time-series. However, hierarchical clustering is essentially not capable to deal effectively with large time-series due to its quadratic computational complexity and accordingly, its use is restricted to small datasets because of its poor scalability.

Partitioning clustering (Non-hierarchical methods)

A partitioning clustering method directly divides n data points into k pre-specified number of clusters without the hierarchical structure, in the way that each group contains at least one object. The k-means clustering is one of the most popular used algorithms, where each cluster has a prototype which is the mean value of its objects. The k means clustering methods seeks an optimal partition of the data by minimizing total distance between all objects in a cluster from their cluster center (prototype), with an iterative optimization procedure. The basic clustering procedure of k-means clustering is summarized

as follows[10]:

1. Initialize a k-partition randomly or based on some prior knowledge. Calculate the cluster prototypes (centroids or means).
2. Assign each unit in the data set to the nearest cluster by using a suitable distance measure between each pair of units and centroids.
3. Recalculate the cluster prototypes (centroids or means) based on the current partition.
4. Repeat steps 2 and 3 until there is no change for each cluster.

Other popular method is k-medoids clustering method or partitioning around medoids (PAM) method, where the units are classified into clusters represented by one of the data points in the cluster; these data points are the prototypes, the so-called medoids. Each medoid synthesizes the cluster information and represents the prototypical features of the clusters and then synthesizes the characteristics of the units belonging to each cluster. The k-medoids clustering method first computes a set of representative units, the medoids. After finding the set of medoids, each unit of the data set is assigned to the nearest medoid units.

For both k-means and k-medoids clustering algorithms, the number of clusters k has to be pre-specified, which is not available or feasible to be determined for many applications. The difficulty obtaining natural clustering results is known as one of their drawbacks. However, k-means and k-medoids are very fast compared to hierarchical clustering which makes them suitable for time-series; indeed, they have been used in many works [1].

In k-means and k-medoids each object is exactly assigned to only one cluster; such hard assignment of objects to clusters can be inadequate in the presence of objects that are almost equally distant from two or more clusters. Fuzzy clustering relaxes the requirement that objects have to be assigned to only one cluster. Objects can belong to more than one cluster and even with different degrees of membership to the different clusters. Fuzzy methods are suitable for clustering time-series for the following motivations:

1. The fuzzy clustering algorithm is attractive because it is a distribution-free procedure [49].
2. Often, is it difficult to identify a clear boundary between clusters in real world problems [49].

3. The fuzzy clustering is computationally more efficient because dramatic changes in the value of cluster membership are less likely to occur in estimation procedures[49].
4. Greater sensitivity in capturing the details characterizing the time-series [49].

Examples of these methods are: Fuzzy c-Means (FcM) clustering, Fuzzy c-Medoids (FcMd) clustering and the Fuzzy c-Means clustering method with noise cluster (FcM-NC). The FcM and FcMd are the same as the methods described before, but the cost function is modified by multiplying the squared distance between the $a-th$ unit and the centroid of the $c-th$ cluster, multiplying this by a term that denotes the membership degree of the $i-th$ object to the $c-th$ cluster. The FcM-NC is a method that adds beside the c clusters to be found in a dataset, the so-called noise cluster; it aims at grouping points that are badly represented by normal clusters, such as noisy data points or outliers.

Model-based methods

Model-based methods attempt to recover the original model from a set of data. This approach assumes a model for each cluster and finds the best fit of data to that model; it assumes that there are some centroids chosen at random and then some noise is added to them with a normal distribution [1]. The model that is recovered from the generated data defines clusters [40]. Model-based methods use either statistical approaches or Neural Network approaches, i.e. Self-Organization Map (SOM), which, given that it needs to define the dimension of weight vector, it cannot work well with time-series of unequal length [22].

Additionally, there are some articles which use model-based clustering of time-series data composed of polynomial models, such as Gaussian mixed models, ARIMA, Markov chain and Hidden Markov models. In general, model-based clustering has two drawbacks: first, it needs to set parameters and it is based on user assumptions which may be false and consequently the result clusters would be inaccurate. Second, it has a slow processing time on large datasets [3]. More detailed information will be given in the Section 2.4 about model-based clustering.

2.1.4. Clustering evaluation measures

The performance of a time-series clustering method must be evaluated with some criteria. Keogh and Kasetty [19] did a research on different articles in time-series mining, in which they conclude that the evaluation of time-series should be performed on various ranges of datasets and new methods of similarity measures should be compared with simple and

stable metrics such as Euclidean distance.

In general, the evaluation of the quality of extracted clusters is not easy in absence of data labels. The definition of the number of clusters, the size of clusters, the definition for outliers, and definition of the similarity measure, are all concepts which depend on the problem at hand and are declared subjectively. Each evaluation criterion has its own benefit and there is no consensus of which criterion is better than other in the data mining community [1]. To evaluate the quality of clustering there are two evaluation techniques: the visualization approach and the scalar measurements.

In scalar accuracy measurements, a single real number is generated to represent the accuracy of different clustering methods, these criteria have values ranging from 0 to 1, where 1 corresponds to the case when ground truth and finding clusters are identical, consequently bigger criteria values are preferred. These numerical measures can judge various aspects of cluster validity and are classified into two types: external index and internal index.

External Index: this index is used to measure the similarity of formed clusters to the externally supplied class labels or ground truth and is the most popular clustering evaluation method [15]. The most common techniques are Rand Index (RI), Adjusted Rand Index, Entropy, Purity, Jacard, F-measure, Folkes and Mallow index (FM), Cluster Similarity Measure (CSM), and MNI. There is no universally accepted technique to evaluate clustering approaches; furthermore, it is generally not directly applicable in real-life unsupervised tasks, because the ground truth is not available for all datasets.

Internal Index: this index is used to measure the goodness of a clustering structure without respect to external information [15]. Internal validity indices evaluate clustering results by using only features and information inherent in a dataset. They are usually used in the case that true solutions (ground truth) are unknown. There are many internal indices such as Sum of Squared Error (SSE), Silhouette index, Calinski-Harabasz, Davies-Bouldin, Dunn index, R-squared index, Hubert-Levin (C-index), Krzanowski-Lai index, Hartigan index, Root-Mean-Square Standard Deviation (RMSSTD) index, Semi-Partial R-squared (SPR) index, Distance between two clusters (CD) index, Homogeneity index, and Separation index. To evaluate clusters in terms of accuracy, the Sum of Squared Error (SSE) is used as the most common measure in different works [1].

2.2. Observation-based approach

The observation-based approach is a direct approach for clustering time-series based on a comparison of the observed time-series or a suitable transformation of the observed time-series [49]. This approach is particularly useful when we want to cluster (univariate or multivariate) time-series based on their geometric profiles and it is also useful when the time-series are not very long.

This approach, as mentioned, works directly with raw data; it employs conventional clustering methods, which are compatible with static data, while the biggest change resides in replacing the distance/similarity measure for static data with an appropriate measure for time-series. To this purpose, in the literature there are various distance measures based on the time observations or their suitable transformations.

All the distances presented in Section 2.1.2 can be suitable for classifying time-series by means of hierarchical and non-hierarchical clustering. In the literature, there are many works based on observation-based clustering models. For clustering multivariate time varying data, Košmelj and Batagelj [20] modified the relocation clustering procedure, originally developed for static data; as similarity measure between trajectories they used Euclidean distance and it works only with time-series of equal length. T.W. Liao [23] applied k-means, fuzzy c-means, and genetic clustering to multivariate battle simulation time-series data of unequal length with the objective of clustering different battle situations. The original time-series data were not evenly sampled and made uniform by using the simple linear interpolation method.

Golay et al. [12] applied the fuzzy c-means algorithm to functional MRI data (univariate time-series of equal length) to provide the functional maps of human brain activity on the application of a stimulus. All three different distances: the Euclidean distance and two cross-correlation based distances were alternatively used in the algorithm. M. Kumar [21] proposed a distance function based on the assumed independent Gaussian models of data errors and used a hierarchical clustering method to group seasonality sequences into a desirable number of clusters.

C. S. Möller-Level [27], in their study of DNA Microarray data, proposed short time-series (STS) distance to measure the similarity in shape formed by the relative change of amplitude and the corresponding temporal information of uneven sampling intervals. By incorporating the STS distance into the standard fuzzy k-means algorithm, they revised the equations for computing the membership matrix and the prototypes (or cluster centers), thus developed a fuzzy time-series clustering algorithm. van Wijk and van Selow

[48] performed an agglomerative hierarchical clustering of daily power consumption data based on the root mean square distance.

Shumway [42] investigated the clustering of non-stationary time-series by applying locally stationary versions of Kullback–Leibler discrimination information measures that give optimal time–frequency statistics for measuring the discrepancy between two non-stationary time-series. To distinguish earthquakes from explosions, an agglomerative hierarchical cluster analysis was performed until a final set of two clusters was obtained.

Vit Niennattrakul and Chotirat Ann Ratanamahatana, for the clustering of multimedia time-series [29] applied k-means and k-medoids algorithms with DTW and demonstrated that k-means is a much more generic clustering method when Euclidean distance is used, but it failed to give correct results when DTW is used. They confirmed that DTW outperforms Euclidean distance metric, and they validated the utility of clustering the time-series data extracted from raw images, instead of clustering these data using traditional image processing techniques. Table 2.1 summarizes the major components used in each raw-data-based clustering algorithm and the type of time-series data for which what designed.

Article	Variable	Length	Representation method	Distance measurement	Clustering algorithm	Evaluation criterion
Košmelj & Batagelj [20]	Multivariate	Equal	Raw time-series	Euclidean (ED)	Modified relocation clustering	Generalized Ward criterion
T.W. Liao [23]	Single	Unequal	Raw time-series	DTW	k-means, fuzzy-k-means, genetic clustering	Within cluster variance
Golay et al. [12]	Single	Equal	Raw time-series	ED and cross correlation	Fuzzy-k-means	Within cluster variance
M. Kumar [21]	Single	Equal	Raw time-series	Independent Gaussian models	Agglomerative hierarchical	N/A
C.Möller Level [27]	Single	Equal	Raw time-series	Short timeseries (STS)	Modified fuzzy k-means	Within cluster variance
Van Wijk & Van [48]	Single	Equal	Raw time-series	Root mean square	Agglomerative hierarchical	N/A
Shumway [42]	Multivariate	Equal	Raw time-series	Kullback-Leibler	Agglomerative hierarchical	N/A
Niennattrakul [29]	Multivariate	Equal	Raw time-series	DTW	k-Mmeans, k-medoids	Within cluster variance

Table 2.1: Summary of observation-based time-series clustering algorithms.

2.3. Feature-based approach

The feature-based metrics are more suitable when the aim is to discriminate between generating processes, that is, between underlying dependence structures [49]. The main concern in this case relies on the global performance so that the features highlighting higher level dynamic aspects could be useful to attain the clustering target [49]. In this approach the clustering target is achieved by using the features extracted, hence a dimensional reduction is attained and consequently computational time saving can be achieved.

In some applications where the series consists of a large number of observations, it is not desirable to work directly with the raw data because of the noise that is present. With the feature-based approach, the raw time-series are converted into a feature vector of lower dimension: the features are extracted in the time domain, the frequency domain and from wavelet decomposition of the time-series. Then, a conventional clustering algorithm is applied to the extracted feature vectors. Usually, in this approach, an equal length feature vector is calculated from each time-series followed by the distance measurement.

Wilpon and Rabiner [53] modified the standard k-means clustering algorithm for the recognition of isolated words with the objectives to develop an automatic clustering algorithm, which could be implemented for any user with a minimal amount of knowledge about clustering procedures, and to provide the template sets as accurate as those created by other clustering algorithms. To measure the distance between two spoken word patterns, a symmetric distance measure was defined.

Shaw and King [41] clustered time-series indirectly by applying two hierarchical clustering algorithms, the Ward's minimum variance algorithm and the single linkage algorithm, to normalized spectra. The spectra were constructed from the original time-series with the means adjusted to zero. The principal component analysis (PCA) filtered spectra were also clustered. The Euclidean distance was used. Fu et al. [11] described the use of self-organizing maps (SOM) for grouping data sequences segmented from the numerical time-series using a continuous sliding window with the aim to discover similar temporal patterns dispersed along the time-series. They introduced the perceptually important point (PIP) identification algorithm to reduce the dimension of the input data.

Goutte et al. [14] clustered fMRI time-series (P slices of images) in groups of voxels with similar activations using two algorithms: k-means and Ward's hierarchical clustering. The cross-correlation function between the fMRI activation and the paradigm (or stimulus) was used as the feature space. Vlachos et al. [50] presented an approach to perform

incremental clustering of time-series at various resolutions using the Haar wavelet transform. Then, the k-means clustering algorithm is applied. Table 2.2 summarizes major components used in each feature-based clustering algorithm. They all can handle series with unequal length because the feature extraction operation takes care of the issue. For a multivariate time-series, features extracted can simply be put together or go through some fusion operation to reduce the dimension and improve the quality of the clustering results.

Article	Variable	Features	Feature selection	Distance measurement	Clustering algorithm	Evaluation criterion
Fu et al. [11]	Single	Perceptually important points	No	Euclidean (ED)	Modified SOM	Expected squared error
Wilpon and Rabiner [53]	Single	LPC coefficients	No	Symmetric measure based on Itakura distance	Modified k-means	Within cluster variance
Shaw & King [41]	Single	Normalized spectra	PCA	Euclidean	Agglomerative hierarchical	N/A
Goutte et al. [14]	Single	Cross correlation function	No	Euclidean	Agglomerative hierarchical & k-means	N/A & Within cluster variance
Vlachos et al. [50]	Single	Haar wavelet transform	No	Euclidean	Modified k-means	Within cluster variance
Owsley et al. [31]	Single	Time frequency representation	Modified SOM	Euclidean	Modified k-means	Within cluster variance

Table 2.2: Summary of feature-based time-series clustering algorithms.

2.4. Model-based approach

In the model-based methods, the time-series under consideration are assumed to have been generated from specific underlying models or by a combination of probability distributions, and the similarity between fitted models is evaluated [52]. So, it is estimated a model from each time-series data, then the data is clustered based on similarity between model parameters.

The most assumed model forms are polynomial mixture models, ARMA models, Markov Chain (MC) and Hidden Markov Models (HMM). However, it is shown that usually model-based approaches have scalability problems, and its performance reduces when the clusters are close to each other [26]. Other drawbacks are that it needs to set parameters based on user assumptions which may be false and consequently the result clusters would be inaccurate; besides, that it has a slow processing time [1].

The earliest works on model-based time-series clustering was that by Piccolo [39] where ARIMA models were fitted to every pair of time-series under consideration, and the Euclidean distance between their corresponding auto regressive expansions were used as the metric. Hierarchical methods were then applied to cluster the time-series. Baragona [5] evaluated three meta-heuristic methods for partitioning a set of time-series into clusters. The cross-correlations were computed from the residuals of the models of the original time-series. Among all methods evaluated, Tabu search was found to perform better than single linkage, pure random search, simulation annealing and genetic algorithms.

Maharaj [24] developed an agglomerative hierarchical clustering procedure that is based on the p-value of a test of hypothesis applied to every pair of given stationary time-series. The clustering result is evaluated with a measure of discrepancy, which is defined as the difference between the actual number of clusters and the number of exactly correct clusters generated. Ramoni [36] presented BCD: a Bayesian algorithm for clustering by dynamics. Given a set S of n numbers of univariate discrete-valued time-series, BCD transforms each series into a Markov chain (MC) and then clusters similar MCs. The similarity between two estimated transition matrices is measured as an average of the symmetrized Kullback–Liebler distance.

K. Kalpakis [16] studied the clustering of ARIMA time-series, by using the Euclidean distance between the Linear Predictive Coding cepstra of two time-series as their dissimilarity measure. The k-medoids algorithm was chosen, with the clustering results evaluated with the cluster similarity measure and Silhouette width. Based on a test of four data sets, they showed that the LPC cestrum provides higher discriminatory power to tell one

time-series from another and superior clustering than other widely used methods such as the Euclidean distance, DFT, DWT, PCA, and DFT.

Xiong and Yeung [54] proposed a model-based method for clustering univariate ARIMA series. They assumed that the time-series are generated by k different ARMA models, with each model corresponding to one cluster of interest. An expectation-maximization (EM) algorithm was used to learn the mixing coefficients as well as the parameters of the component models. The proposed method was compared with that of Kalpakis et al. [16] using the same four datasets.

Beran and Mazzola [7] defined hierarchical smoothing models (or HISMOOTH models) to understand the relationship between the symbolic structure of a music score and its performance, with each represented by a time-series. The models are characterized by a hierarchy of bandwidths and a vector of coefficients.

Oates et al. [30] presented a hybrid clustering method for automatically determining the k number of generating HMMs, and for learning the parameters of those HMMs. A standard hierarchical, agglomerative clustering algorithm was first applied to obtain an initial estimate of k and to form the initial clusters using dynamic time warping to assess the similarity. These initial clusters serve as the input to a process that trains one HMM on each cluster and iteratively moves time-series between clusters based on their likelihoods given the various HMMs. Table 2.3 summarizes major components used in each model-based clustering algorithm.

Article	Variable	Model	Model output of interest	Distance measurement	Clustering algorithm	Evaluation criterion
Piccolo [35]	Single	ARIMA	Coefficients	Euclidean	Agglomerative hierarchical	N/A
Baragona [5]	Single & Multivariate	ARMA	Residuals	Cross correlation based	Tabu search, GA & simulated annealing	Specially designed
Maharaj [24]	Single	AR Coefficients	P-value of hypothesis testing	Specially designed	Agglomerative hierarchical	N/A
Ramoni et al. [36]	Single	Markov Chain	Transition probabilities	Kullback-Liebler	Agglomerative hierarchical	N/A
K. Kalpakis [16]	Single	AR	LPC coefficients	Euclidean	Partition around medoids	Cluster similarity metric & Silhouette width
Xiong & Yeung [54]	Single	ARMA mixture	Coefficients	Log-likelihood	Expectation maximization EM learning	Cluster similarity metric
Beran & Mazzola [7]	Single	Hierarchical smoothing models	Coefficients	Euclidean	Agglomerative hierarchical	N/A
Oates et al. [30]	Multivariate	Discrete HMM	HMM parameters	Log-likelihood	Initialized by DTW followed by a fixed point operation	Log-likelihood

Table 2.3: Summary of model-based time-series clustering algorithms.

3 | The Proposed Timex-Clustering Framework

The objective of this work is to develop an automatic clustering framework, which could be used by any user on their datasets with a minimal amount of knowledge about clustering procedures. It should provide efficient and high quality time-series clustering, as accurate as those created by other clustering algorithms. An analogous innovation, but in the field of time-series forecasting, is TIMEX, created by Alessandro Falcetta [2], it provides in an "as-a-service" manner, a fully automatic end-to-end forecasting pipeline.

This thesis introduces TIMEX-CLUSTERING, an open-source Python-based framework for time-series clustering-as-a-service. Differently from the tools and libraries present in the literature, which provide excellent results only for specific cases, TIMEX-CLUSTERING provides in an "as-a-service" manner, a fully automatic end-to-end clustering pipeline comprising all the steps involved to cluster (such as data ingestion, data preprocessing, data description, data clustering and service delivery).

As shown in the vast volume of time-series literature, traditional time-series analysis and modeling tend to be based on non-automatic and trial-and-error approaches; therefore, to achieve an efficient and high-quality time-series clustering, the first challenge is to understand the unique characteristics of the subject data and then to choose an appropriate similarity measure accordingly. The second challenge is to accomplish the clustering in a fully automatic end-to-end pipeline comprising the different steps involved according on the clustering approach chosen, also with the final objective of offering all the relevant characteristics that an as-a-service framework should have.

As pointed by T. Warren Liao [52], nowadays there is a lack of studies to compare different time-series clustering approaches. For that reason, and to solve the first challenge previously declared, TIMEX-CLUSTERING will use the three different clustering approaches (observation-based, feature-based and model-based) in order to cover datasets with different nature. It is emphasized that a clustering-as-a-service is not yet available in the literature: this framework will be also unique because will include all the clustering

approaches.

The proposed TIMEX-CLUSTERING framework, developed in Python, is characterized by the pipeline depicted in Figure 3.1. Such pipeline comprises the following six steps: data ingestion, data pre-processing, data description, data clustering and service delivery.

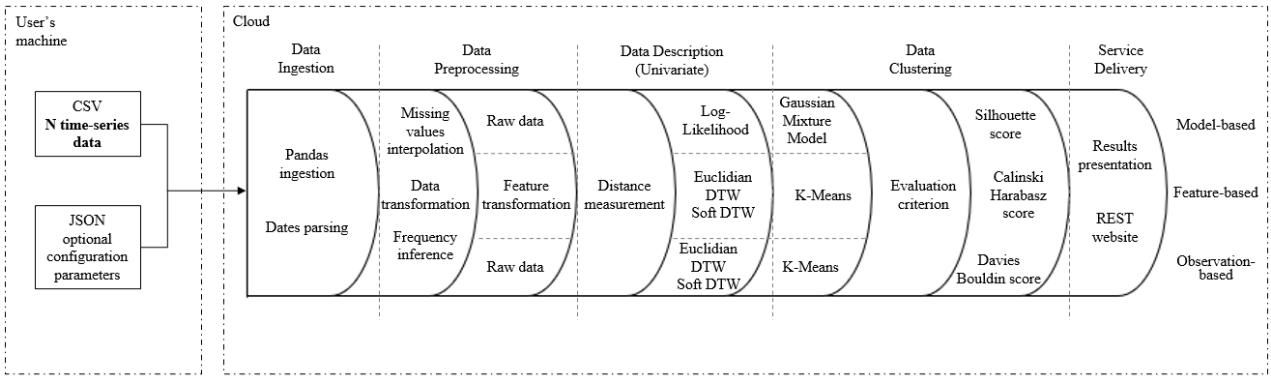


Figure 3.1: The pipeline of the proposed TIMEX-CLUSTERING framework.

Each step in the clustering pipeline and its corresponding autonomy will be explained in the next subsections, but at great features each step consists of:

- Data ingestion: the input data received is parsed and transformed into a computationally effective format, Python Pandas [41] data structures.[32].
- Data pre-processing: missing data are identified and handled (by means of interpolation), and the parsed data is modified by feature transformations to emphasize specific aspects if needed according to the clustering approach used.
- Data description: choosing an adequately accurate distance measure is controversial, therefore based on the specific feature transformations applied, a similarity measure is used to calculate the distance among univariate time-series.
- Data clustering: according to the clustering approach used the corresponding clustering methods are applied, and it is identified the cluster model with the largest accuracy by exploring the different clustering algorithms and different algorithm configurations.
- Service delivery: the outcomes of the clustering are made available in a ready-to-use form through API or web console.

As its homologous framework [2], TIMEX-CLUSTERING works in an "as-a-service" approach: it processes automatically from end-to-end the user's input data through all the clustering steps without requiring human intervention. The TIMEX framework, which

is made available through API or web-console, receives as input a CSV file storing the N univariate time-series used to cluster the data and (optionally) a JSON file storing the parameter configuration. Besides, TIMEX-CLUSTERING is released to the scientific community as an open-source framework and can be easily ported to different computing infrastructures. Although the framework is intended to be use in a fully automatic way (which means that each step is automatically computed, without user intervention), users can also take advantage of single parts of the framework.

3.1. Assumptions and scopes of the framework

As previously mentioned, although many different researches have been conducted on time-series clustering, in order to achieve an efficient and high-quality time-series clustering some assumptions are required: the unique characteristics of the data influence the choice of the representation method, the similarity measure, and the clustering algorithm. The assumptions explained underneath specify the limitations and scopes of this thesis.

The focus of this thesis is on whole time-series clustering, which means applying conventional clustering on discrete objects (a set of individual time-series). This scope is chosen because sub-sequence clustering or time point clustering performed on a single time-series is meaningless as represented by Keogh and Lin [18].

A very strong assumption in this study is to process only univariate time-series. This assumption is taken because calculating distance between two time-series by considering them as univariate time-series is one of the simplest ways to do it [1], for the reason that by considering them as univariate and then calculating the distance measurement across all time points, computational cost and complexity of comparing two time-series is saved. Although some constraints or considerations in the similarity measurements can be taken to mitigate the complexity of comparing multivariate time-series, it needs careful tuning of parameters to be efficient and effective; for that reason multivariate time-series are out of the scope of this thesis. As a result, the final goal of this study is to find a trade-off between speed and accuracy by using the most adequate distance metrics for univariate time-series. A future complementary work for this framework, it is to add the functionalities for clustering multivariate time-series of equal or unequal length by applying e.g. k-means or fuzzy k-means clustering algorithm to time stripped data to convert multivariate real-valued time-series into univariate discrete-valued time-series as Liao [22], because multivariate time-series, features can improve the quality of the clustering results.

The univariate time-series considered are discrete-valued and uniformly sampled, all with

equal length. In case that the time-series are non-uniformly sampled or have unequal length, the data is converted into uniformed and equal length data before clustering operations by using the simple linear interpolation method.

The final assumption is regarding the clustering methods that will be used, remaining that there exist five categories of clustering methods for handing static data: partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods. In this work only three of the five major categories of clustering methods for static data, specifically partitioning methods, hierarchical methods, and model-based methods, will be employed directly or modified for time-series clustering, because grid based and density-based algorithms usage are scarce for the same problem of slow processing and high complexity.

Partitioning algorithms are widely used because of their fast response. However, they have as a drawback that the number of clusters needs to be pre-assigned; in addition, because of their dependency to prototypes, they are more suitable for clustering equal length time-series. Hierarchical clustering, on the other hand, does not need the number of clusters to be predefined, it has a great visualization power in time-series clustering, it can handle cluster time-series with unequal length and is a perfect tool for evaluation of dimensionality reduction or distance metrics. But hierarchical clustering is restricted to the small data sets because of its quadratic computational complexity. Concerning the model-based methods, the time-series under consideration are assumed to have been generated from specific underlying models, subsequently the data is clustered based on similarity between the estimated model parameters of each time-series data.

3.2. Data ingestion

The data ingestion step is activated by the user through API or a web console. Two inputs are expected:

- a dataset, in CSV or JSON format, storing the M number of univariate time-series. If the dataset is available as an online resource, its URL can be specified in the JSON configuration file and TIMEX-CLUSTERING will download it automatically.
- (optional) a configuration file, in JSON format, detailing the parameters that are available in the TIMEX-CLUSTERING. When the configuration file is not specified by the user, a default configuration file is used. An example of such a configuration file is given in Appendix A for the Covid-19 application scenario.

The first column stores the timestamps of the time-series in the ISO 8601 format, while

the remaining M columns store the M time-series. The first row, called header row, comprises the names of the columns, i.e., the names of the time-series (e.g., temperature, humidity, light, etc.). The remaining N rows store the values of the M time-series at the corresponding timestamps. Without any loss of generality the M time-series are assumed to be synchronous with the same sampling frequency. Missing or unevenly sampled data are managed in the next step.

The dataset is imported in TIMEX-CLUSTERING through the Pandas [32] and NumPy [47] Python libraries and made available to the next steps in the form of a Pandas DataFrame, called TS_DataFrame. Timestamps are automatically parsed by using the *dateparser*2 library. An example of a Pandas DataFrame for TIMEX-CLUSTERING is given in *Chapter 4* for the Covid-19 use case. Similarly, the configuration file, which is imported through the standard Python JSON-import function, is made available in a Python dictionary form, called $Config\_Dict$.

3.3. Data pre-processing

The data pre-processing step, which receives in inputs TS_DataFrame and $Config\_Dict$ developed by the data ingestion step, aims at transforming the time-series present in TS_DataFrame into a form that data description and data clustering can easily parse.

More specifically, the data pre-processing step is organized into three different phases: handling missing data, data pre-transformation and feature transformation. The phases are fully automatic in the TIMEX-CLUSTERING pipeline.

3.3.1. Missing data

This first phase aims at identifying and correcting possible missing data present in the TS_DataFrame. For this purpose the periodicity of the time-series is automatically estimated from data by identifying the minimum sampling period, representing the estimated periodicity of the time-series. This analysis is necessary because the samples of the time-series may not have been collected regularly. After estimating the periodicity, missing data are detected and handled by means of the interpolation mechanisms that are natively available in Pandas.

3.3.2. Data pre-transformation

Once missing data have been handled, the data streams are modified through data transformation techniques to make data more understandable for the next clustering steps.

Three different data transformation techniques are available in TIMEX-CLUSTERING, the first is a scaler transformation which output data have zero mean and unit variance. The other two transformations make the time-series stationary, a time-series is stationary if its statistical properties do not vary over time [28]. Stationarity is often an (underlying) assumption of specific families of clustering models. For this purpose and to manage exponentially-increasing time-series, TIMEX-CLUSTERING relies on the approximated logarithmic transformation and on the following modified logarithmic transformation:

$$\tilde{log}(x) = sign(x) \cdot log(|x| + 1) \tag{3.1}$$

This approximation is meant to overcome the issues of the regular logarithm in case of zero or negative values. The selection of the specific data transformation technique to use (or not) is not made automatically by the framework, and it is up to the user to decide whether to use it or not, according to the results obtained, as can be seen in the examples in *Chapter 4*.

3.3.3. Feature transformation

For the Feature based clustering approach, as explained in chapter 2, a feature transformation is applied to the data to reduce the dimensionality of the time-series. As seen in subsection 2.1.4 time-series data are frequently large and may contain outliers, and the feature transformation or representation method is a key component to solve this issue.

Each time-series dimensionality reduction technique has its own features and drawbacks; the representation technique that will be used is DWT because it can be used for stationary and non-stationary time-series. Also, it is suitable for time-series with fix size (equal-length) segmentation, which make the comparison of representations of several time-series straightforward. The fix size segmentation is achieved by the data preprocessing in subsection 3.3.1.

Using the DWT method will reduce the dimensionality of the data in half, significantly saving computation time, as it will be seen in *Chapter 5*. The dimensionality reduction could be smaller by performing a multilevel decomposition, but it was considered that a smaller reduction could lead to a loss of information, so only the approximation coefficients of the first level are used. It was also decided to use Wavelet transform because it provides a multi-resolution representation; the wavelet used was Haar wavelet. The preference for the Haar wavelet is mainly based on its simplicity and its wide usage in the data mining community; although the Haar wavelet was chosen, the algorithm can generally utilize any wavelet basis (Daubechies, Gaussian wavelets, Mexican hat wavelet, etc.)

3.4. Data description

3.4.1. Distance/Similarity measures

In this section, the different distance/similarity measures chosen to measure the distance among the time-series will be explained. As explained in the Section 3.1, the time-series considered are univariate time-series, which make simpler the computation of the distance across all time points of two time-series.

The choice of an adequate accurate distance measure is a trade-off between speed and accuracy. The most effective and accurate approaches are the ones which are based on dynamic programming but they are expensive in time execution (the cost is quadratic in the length of the time-series).

Observation and Feature based approaches distance metrics

For the Observation based and Feature based clustering approaches it was decided to use the following metrics: Euclidean distance, DTW and soft DTW. The Euclidean distance will allow to find similar time-series in time; this kind of distance metric is costly on raw long length time-series, so, for long time-series it is recommended to compute the distance on transformed time-series by using DWT available in the Feature based approach.

To find similar time-series in shape where the time of occurrence of patterns is not important, the Dynamic Time Warping (DTW) and soft DTW method, was chosen to perform the distance measurement. These metrics allow a non-linear mapping of two vectors by minimizing the distance between them. The main drawback is that it requires more computational effort in comparison with Euclidean distance, however, it is robust to shifts or dilatations across the time dimension. One strong limitation of DTW is that it cannot be differentiated everywhere because it is solved by minimal-cost alignment that is used throughout the computations. Contrary to DTW, soft-DTW is differentiable everywhere, mitigating the DTW limitation. The formal definition for soft-DTW[9] is the following:

$$soft{-}DTW^{\gamma}(x, x') = \min_{\pi \epsilon A(x,x')}^{\gamma} \sum_{(i,j)\epsilon\pi} d(x_i, {x_j}')^2 \tag{3.2}$$

where $\min^{\gamma}$ is the soft-min operator parametrized by a smoothing factor γ, that can be chosen by the user or is set automatically by the framework.

Model based approach distance metric

In the Model based clustering approach it is not used a distance metric, instead, the Gaussian mixture model uses the Expectation-maximization algorithm to find which cluster each time series comes from. The EM algorithm first assumes random components (randomly centered on time-series, learned from k-means, or even just normally distributed around the origin) and computes for each time-series a probability of being generated by each component of the model. Then, it tweaks the parameters to maximize the likelihood of the data given those assignments. Repeating this process is guaranteed to always converge to a local optimum. So the distance metric used is the likelihood.

3.4.2. Univariate data description

The goal of the data description step is to extract the relevant information from the M time-series stored in PP_DataFrame, characterizing the relationships among all the M time-series. The information provided by this step comprises:

- plots and graphs, which will be returned to the user as part of the TIMEX result. These help the user in understanding the temporal behavior of the time-series,
- a dependency graph modeling the relationship among the M time-series.

Through the data description phase graphical insights about all the time-series are provided. Following the "as-a-service" approach, these graphical insights are automatically computed by TIMEX-CLUSTERING and saved in PNG format, that are then stored into a ZIP message to be returned to user through REST, or included into a dynamically created webpage showing the results. In TIMEX-CLUSTERING, the following univariate data-description techniques are available:

- line plot to show the time evolution of the data and highlight possible trends or seasonality behavior,
- a dependency graph to characterize the relationships among the time-series, explained below,
- a cross-correlation plot for each time-series is made available to the user to graphically depict the behavior of the cross-correlation of each time-series with all the other time-series.

The dependency graph is a directed graph $G = \{M, C\}$, where M is the set of M nodes, each of which representing a time-series, and C is the set of connections between nodes in M. Being m_i and m_j the nodes corresponding to the $i-th$ and $j-th$ time-series,

respectively, a connection $c_{i,j}$ exists between m_i and m_j if the absolute value of the peak of the cross-correlation between time-series i and j is above a predefined threshold γ (tunable parameter within the TIMEX-CLUSTERING). TIMEX-CLUSTERING have three algorithms for the cross-correlation computation: Pearson [6], Kendall [17] and Spearman [43]. The cross-correlation is computed in the range of time-lags $[-N/2, +N/2]$, being N the length of the time-series in the dataset.

Examples of the usage of these techniques in the COVID-19 pandemic use case are given in *Chapter 5*.

3.5. Data clustering

3.5.1. Clustering methods

The core of TIMEX-CLUSTERING is the ability to provide users an automatic tool to perform with high efficiency and quality time-series clustering. For this purpose, TIMEX-CLUSTERING supports three different clustering approaches, with a varied range of time-series clustering models present in the clustering library ϕ_{TIMEX}. Currently the models available in ϕ_{TIMEX} are: K-Means (for the Observation and Feature based approach) and Gaussian Mixture model (for the Model based approach), but the framework can be easily extended to include new models. It is worth noting that, inspired by the "as-a-service" approach of TIMEX-CLUSTERING, the user is not required to specify the clustering models to use nor having expertise in the field. The data clustering step is fully automatic.

For this purpose, TIMEX-CLUSTERING leverages an iterative approach aiming at identifying the best clustering model by exploring:

- the different families of clustering models available for each one of the three clustering approaches in ϕ_{TIMEX},
- the automatic search of the best number of clusters for the data, among the number of clusters suggested by the user or by defining the most suitable set of number of clusters where to search on,
- the automatic definition of the best model: best number of clusters and best distance/similarity metric, for each one of the clustering approaches specified, and according to the validation metric specified, if none validation metric is specified the framework will use the most suitable validation metric to evaluate the performance of the clustering algorithms.

There are different ways of performing clustering, as mentioned in *Chapter 2*, in the below subsections it will be explained the algorithms used for each clustering approach.

For the dataset ingested, TIMEX-CLUSTERING learns the univariate clustering models available in ϕ_{TIMEX}, the procedure is applied using one distance metric at a time for each model and the corresponding evaluation metrics are computed. At the end, the configuration τ of:

- clustering approach and clustering model in ϕ_{TIMEX},
- number of clusters considered,
- distance metrics considered,
- feature transformation considered,
- data pre-transformation considered,

providing the largest score according to the evaluation metric is selected. It is emphasized that all the tasks in this step are completely automatic masking the complexity of the pipeline infrastructure and the expertise required to define a time-series clustering model. The output of the data clustering step comprises:

- the best clustering model for each clustering approach; plots detailing the clusters distribution, subplots for each distance metric showing the cluster centers with the time-series that belongs to that cluster,
- a data frame indicating the cluster to which each time-series belong,
- a vector with the cluster indexes to which each time-series belong,
- the configuration τ providing the largest evaluation metric score.

Observation and Feature based clustering algorithm

For these clustering approaches the algorithm use is *k-means*, implemented by using Scikit-learn [34] and Ts-learn [44] Python libraries, which have many machine learning algorithms available.

It was decided to use a non-hierarchical method, because the hierarchical methods are weak in terms of quality since they cannot adjust the new clusters after splitting or merging previous clusters. From the non-hierarchical algorithms, it was decided to use the *k-means* method. *K-means* clustering is one of the best-known and most popular clustering methods; this algorithm requires the number of clusters k to be specified. The average complexity is given by $O(k \cdot n \cdot T)$, where n is the number of samples and T is the

number of iterations, it was opted to use k-means because in practice, *k-means* algorithm is very fast (one of the fastest clustering algorithms available). Within its disadvantages, *k-means* does not have a great visualization power in time-series clustering as hierarchical algorithms; however, *k-means* is capable to deal effectively with large time-series, big difference with respect to hierarchical algorithms that have a quadratic computational complexity and accordingly, it leads to be restricted to small datasets because of its poor scalability.

The *k-means* algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares, equation (3.3). This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields. [34]

$$\sum_{i=0}^{n} \min_{u_j \epsilon C}(||(x_i - u_j)||^2) \tag{3.3}$$

The most common challenge with *k-means* is that it must know the clusters beforehand: it cannot learn the number of clusters from the data, however the most common approach that is rather intuitive, is the is called silhouette analysis by using the silhouette score available in TIMEX-CLUSTERING. Alternatively, it might be used a more complicated clustering algorithm which has a better quantitative measure of the fitness per number of clusters (e.g., Gaussian mixture models, used in the Model based approach) or use an algorithm that chooses the suitable number of clusters (e.g., DBSCAN, mean-shift, or affinity propagation).

The main drawbacks are that due to fundamental model assumptions of *k-means* (points will be closer to their own cluster center than to others), the algorithm will sometimes be ineffective if the clusters have complicated geometries. Also because each iteration of k-means must access every time-series in the dataset, the algorithm can be relatively slow as the number of samples grows.

Model based clustering algorithm

For the Model based approach the algorithm used is *Gaussian mixture model*, implemented by using Scikit-learn [34] Python library. A *Gaussian mixture model* is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters.

The *k-means* clustering model explored in the previous section is simple and relatively easy

to understand, but the non-probabilistic nature of k-means and its use of simple distance from cluster-center to assign cluster membership sometimes lead to poor performance. *Gaussian mixture models* (GMMs), can be viewed as an extension of the ideas behind *k-means*.

The two big disadvantages of *k-means* are its lack of flexibility in cluster shape and lack of intrinsic measure of probability or uncertainty of cluster assignment, meaning that for many datasets (especially low-dimensional datasets) it may not perform well. A *Gaussian mixture models* (GMM) attempts to find a mixture of multi-dimensional Gaussian probability distributions that best models any input dataset. In the simplest case, *GMMs* can be used for finding clusters in the same manner as k-means; but because *GMM* contains a probabilistic model under the hood, it is also possible to find probabilistic cluster assignments. Similarly, the *GMM* approach can be used to fit the stretched dataset; allowing for a full covariance the model will fit even very oblong, stretched-out clusters. This makes clear that *GMM* addresses the two main practical issues with *k-means* encountered before.

The fact that *GMM* is a generative model gives a natural means of determining the optimal number of components for a given dataset. A generative model is inherently a probability distribution for the dataset, and so we can simply evaluate the likelihood of the data under the model, using cross-validation to avoid over-fitting. Another means of correcting for over-fitting is to adjust the model likelihoods using some analytic criterion such as the Akaike information criterion (AIC) or the Bayesian information criterion (BIC) [34].

The main two drawbacks of *GMMs* are that it needs to set initial parameters (number of components to use), and that when there are not sufficiently points per mixture, estimating the covariance matrices becomes difficult: the algorithm is known to diverge and find solutions with infinite likelihood. However, as this algorithm maximizes only the likelihood, it will not bias the means towards zero, or bias the cluster sizes to have specific structures that might or might not apply.

3.5.2. Clustering evaluation measures

Evaluating the performance of a clustering algorithm is not as trivial as counting the number of errors or the precision and recall of a supervised classification algorithm. In particular any evaluation metric should not take the absolute values of the cluster labels into account but rather if this clustering define separations of the data similar to some ground truth set of classes or satisfying some assumption such that members belong to the same class are more similar than members of different classes according to some similarity

metric. Each evaluation criterion has its own benefit and there is no consensus of which criterion is better than other in the data mining community [1].

To evaluate the performance of the time-series clustering methods in TIMEX-CLUSTERING, it was assumed that the knowledge of the ground truth class assignments is unknown and not given by the user, because usually in real-life unsupervised tasks the ground truth is not available. Hence, TIMEX-CLUSTERING evaluates the performance of the clustering methods using three different internal index measures, as explained in subsection 2.1.4. Internal validity indices evaluate clustering results by using only features and information inherent in a dataset; the three indices available in TIMEX-CLUSTERING are: Silhouette index, Calinski-Harabasz, Davies-Bouldin.

The framework will automatically defined the best model for each one of the clustering approaches according to one of these three evaluation metrics, chosen by the user or automatically defined if not specified by the user.

Silhouette index

If the ground truth labels are not known, evaluation must be performed using the model itself. The Silhouette Coefficient is an example of such an evaluation, where a higher Silhouette Coefficient score relates to a model with better defined clusters [34]. The Silhouette Coefficient is defined for each sample and is composed of two scores:

- a: The mean distance between a sample and all other points in the same class.
- b: The mean distance between a sample and all other points in the next nearest cluster.

The Silhouette Coefficient s for a single sample is then given as:

$$s = \frac{b - a}{\max(a, b)} \tag{3.4}$$

The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters. The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster. The Silhouette Coefficient is generally higher for convex clusters than other concepts of clusters, such as density based clusters [34].

Calinski-Harabasz index

If the ground truth labels are not known, the Calinski-Harabasz index, also known as the Variance Ratio Criterion, can be used to evaluate the model, where a higher Calinski-Harabasz score relates to a model with better defined clusters.

The index is the ratio of the sum of between-clusters dispersion and of within-cluster dispersion for all clusters (where dispersion is defined as the sum of distances squared). For a set of data E of size n which has been clustered into k clusters, the Calinski-Harabasz score is defined as the ratio of the between-clusters dispersion mean and the within-cluster dispersion [34].:

$$s = \frac{tr(B_k)}{tr(W_k)} - \frac{n_E - k}{k - 1} \tag{3.5}$$

where $tr(B_k)$ is trace of the between group dispersion matrix and $tr(W_k)$ is the trace of the within-cluster dispersion matrix. The Calinski-Harabasz index is generally higher for convex clusters than other concepts of clusters, such as density based clusters like those obtained through DBSCAN [34].

Davies-Bouldin

If the ground truth labels are not known, the Davies-Bouldin index can be used to evaluate the model, where a lower Davies-Bouldin index relates to a model with better separation between the clusters. This index signifies the average 'similarity' between clusters, where the similarity is a measure that compares the distance between clusters with the size of the clusters themselves. Zero is the lowest possible score. Values closer to zero indicate a better partition [34].

The Davies-Boulding index is generally higher for convex clusters than other concepts of clusters, such as density based clusters. The computation of Davies-Bouldin is simpler than that of Silhouette scores, however the usage of centroid distance limits the distance metric to Euclidean space [34].

3.6. Service delivery

The TIMEX-CLUSTERING framework can be delivered in two different modes: REST API and a website accessible with a browser.

3.6.1. REST endpoint

TIMEX-CLUSTERING exposes a REST API, built with the Python library Flask. The client (any application able to perform REST requests) has to invoke the TIMEX-CLUSTERING endpoint with a POST request, whose payload shall contain the CSV/JSON dataset file and, optionally, the JSON configuration file. Once received, TIMEX-CLUSTERING will activate the clustering pipeline described above. At the end of the pipeline, TIMEX-CLUSTERING creates a ZIP that is returned to the user as attachment to the POST request's answer; the ZIP file comprises the following results:

- the plots and figures (in PNG format) delivered by the data description and data clustering steps;
- a JSON file containing the clusters computed in the data clustering phase;

3.6.2. Website

TIMEX-CLUSTERING can also be delivered through a clustering as-a-service website, where users can upload the CSV/JSON dataset file and the optional JSON configuration file, and results are provided by means of the open-source Python library Dash3. It was opted to use Dash since, operating on top of Flask, Plotly.js, and React.js, it allows the creation of interactive web pages supporting the upload of files and the visualization of plots and graphs. In principle different data visualization tools or libraries can be considered. One of the main advantages of using a website for providing the service is that web browsers are cross-platform and widely diffused, making the service accessible to a large audience. An example of the usage of TIMEX-CLUSTERING website can be seen in the Figure 3.2 and it will be described with detail in *Chapter 5*.

Figure 3.2: Example of the clustering as-a-service website.

4 | Data

In this chapter, it will be showed the datasets used in the creation of TIMEX-CLUSTERING and in the different experiments that were performed. For each dataset it is given a description of its source, general information of them, their creation process and the reasons why those datasets were chosen to show the characteristics and performance of the TIMEX-CLUSTERING framework.

The TIMEX-CLUSTERING framework for time-series clustering-as-a-service was applied to two datasets, the clustering of the COVID-19 pandemic new cases spread worldwide and the clustering of a distinct dataset extracted from the UCR & UEA repository [4], the most recognized repository of the most influential research surrounding the subject of time-series classification and clustering. In the last section, a summary is as a preamble to the *Chapter 5* referring to the clustering results obtained.

4.1. UCR & UEA Trace dataset

As mentioned, the source of this dataset was the UCR & UEA repository [4], specifically is the dataset named *Trace*, which is a synthetic dataset designed to simulate instrumentation failures in a nuclear power plant, created by Davide Roverso [39]. It is a 4 class dataset, subset of the Transient Classification Benchmark (trace project), an initiative at the turn of the century to collate data from the application domain of the process industry (e.g. nuclear, chemical, etc.). As explained in the Section 2.1, finding the clusters in this kind of time-series can help to solve real world problems, as in this example it can assist to identify anomalies or patterns in the sensor databases of industrial processes.

This dataset contains 200 instances, 50 for each class with a length of 275 data points. From this dataset it was removed one class, finishing with a total of 3 classes, the reason of this was to have results and plots more comprehensible for this work, however the TIMEX-CLUSTERING framework can cluster any input dataset. The purpose to use this synthetic dataset, with known ground truth classes, was to validate the correct performance of each clustering algorithm within the proposed framework. The number of

time-series was reduced to 50 samples, and the length was also reduced to 40 data points, the reason of this was to confirm the hypothesis that the observation based approach achieved a higher quality clustering for time-series of short length in comparison with the model and feature based clustering approaches.

Another reason to choose this dataset is that, as it can be seen in the Figure 4.1, the transient behavior of the three distinct classes will allow us to analyze the performance of each one of the three distance/similarity metrics available: Euclidean, DTW and soft-DTW, it will show us the advantages and drawbacks of each one of them when clustering the time-series.

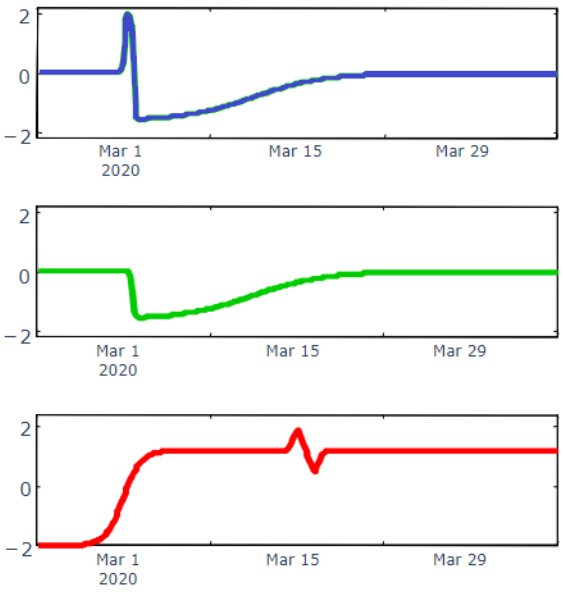


Figure 4.1: Transient behaviour cases.

Moreover in the Figure 4.1, it can be perceived that the first class (blue line) has a behavior that can be distinguished from the second class (green line) by the presence of a high frequency peak at the beginning of the transient, while the third class (red line) has a behavior with a small burst in the middle phase of the transient. The first two classes could be attributed to two different behaviors of a unique sensor, exemplifying the presence of the peak as an anomaly in the measurement of that sensor, while the third class could be attributed to a sensor measurement of a different variable. It could be applied a multivariate clustering considering the combination of the measurements of various sensors as a multivariate time-series input, attributing each combination to a specific state of an industrial process; however, multivariate clustering of time-series is beyond the scope of this work.

Finally, it can be appreciated in the Figure 4.2 the time-series ingested using the tool

available in TIMEX-CLUSTERING, through the time evolution of the data. By analyzing the dimensions in general of all the time-series it can be concluded that there is no need no apply a pre-transformation to the data as the logarithmic or modified logarithmic explained in subsection 3.3.2. It could be applied a scale of the series by using the pre-transformation available; however, by looking at the time-series, the scale is not so different among them. Towards the end it will not affect the accuracy of the clustering process.

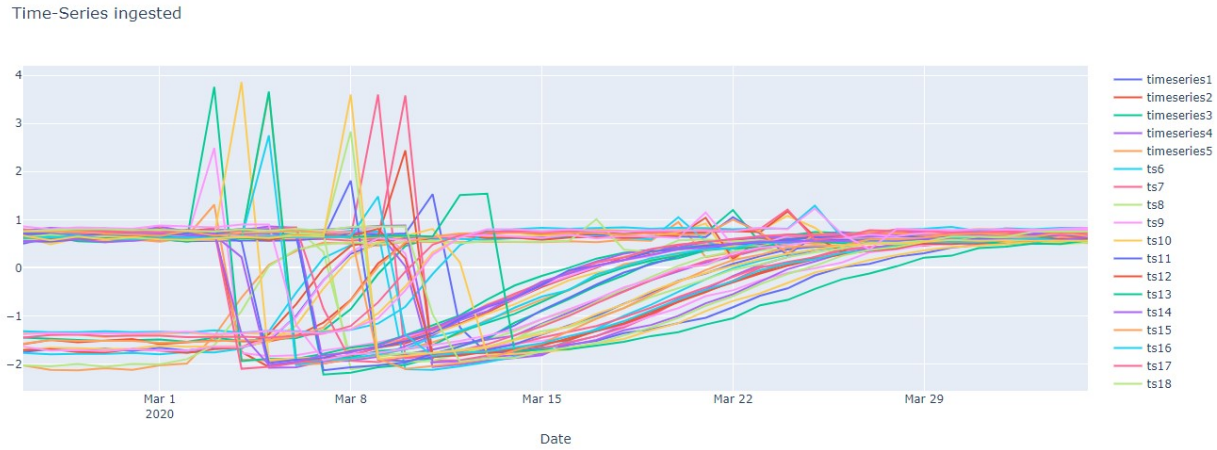


Figure 4.2: Time-series ingested from the Trace dataset.

4.1.1. Experiments' structure Trace dataset

The structure of the experiments for this dataset will be the following: it will be applied the three clustering approaches, observation, feature and model based, to compare them with each other, analyzing advantages and drawbacks of each one of them. The clustering algorithms applied will be K-Means (observation and feature based approaches) and Gaussian Mixture model (model based approach). The distance metrics used will be Euclidean, DTW and soft-DTW, with an analysis of the advantages and drawbacks of each metric. Results of the clustering with and without a data preprocessing transformation will be presented, and for the Feature based approach the feature transformation used will be DWT using the Haar wavelet. It will be performed an exhaustive search over the specified parameter configurations and over a range from 3 to 6 number of clusters. Table 4.1 summarizes the major components used for the experiment on the Trace dataset.

Clustering approach	Variable	PreTrans-formation	Distance metrics	Feature repre-sentation	Model	Evaluation criterion
Observation based	Univariate	None & Log-Modified	Euclidean DTW softDTW	N/A	K-Means	Silhouette score
Feature based	Univariate	None & Log-Modified	Euclidean DTW softDTW	DWT, Haar wavelet	K-Means	Silhouette score
Model based	Univariate	None & Log-Modified	Euclidean DTW softDTW	N/A	Gaussian Mixture Model	Silhouette score

Table 4.1: Structure of the experiments for the UCR & UEA Trace dataset.

4.2. COVID-19 dataset

The source of this dataset is the website Our World in Data [38] which allows to explore the evolution and statistics on the coronavirus pandemic for every country in the world. This repository focuses on data related to the main problems in the world: poverty, disease, hunger, climate change, war, existential risks, inequality, etc., and it has a specific section associated to all the data accumulated during the coronavirus pandemic.

Centering on the pandemic information, the repository updates daily the data on the coronavirus pandemic. It counts with 207 country profiles which allows one to explore the progression and statistics for every country. Although the COVID-19 is a fast-evolving pandemic, with this dataset the goal is to identify the most significant homogeneous groups of countries, which could lead us to get a more comprehensive assessment of the impact of the pandemic and discover patterns useful to handle a pandemic by identifying the countries that were most successful in making progress against it.

From the 207 country profiles of the COVID-19 dataset, TIMEX-CLUSTERING will process only the data related to the daily cases (new positive cases) of each country, this is the typical number reported by media. It could be done a multivariate clustering by considering more variables of the data pandemic as total positives, total hospitalized, total intensive care, etc., however, multivariate clustering of time-series is beyond the scope of this work. From the 207 country profiles, only the 193 countries of the United Nations

were initially considered, then, from those countries, only the countries with at least half of the data within the time period explored were considered, finishing with 181 countries. The time period studied covers from February 24, 2020 to February 20, 2022; which is a total length of 728 daily samples for each country.

One of the purposes to use this real-world scenario dataset, is that the ground truth labels are not known, which is a common situation in real-life unsupervised tasks. So in this way, it can be evaluated how well TIMEX-CLUSTERING performs in real case scenarios by evaluating the efficiency and quality of the clustering obtained; in the next *Chapter 5* it will be analyzed with detail the most relevant insights and results provided by TIMEX-CLUSTERING on the COVID-19 pandemic case.

With the COVID-19 dataset will be possible to validate a different hypothesis than the one established for the UCR & UEA dataset; as in COVID-19 dataset the number of time-series is bigger than in the previous dataset, and also the length of each time-series is longer, for this dataset the hypothesis is that by using the Model and Feature based clustering approaches, we will obtain clusters more efficiently (saving computational time) and with higher quality than the clusters that could be obtained with the Observation based approach.

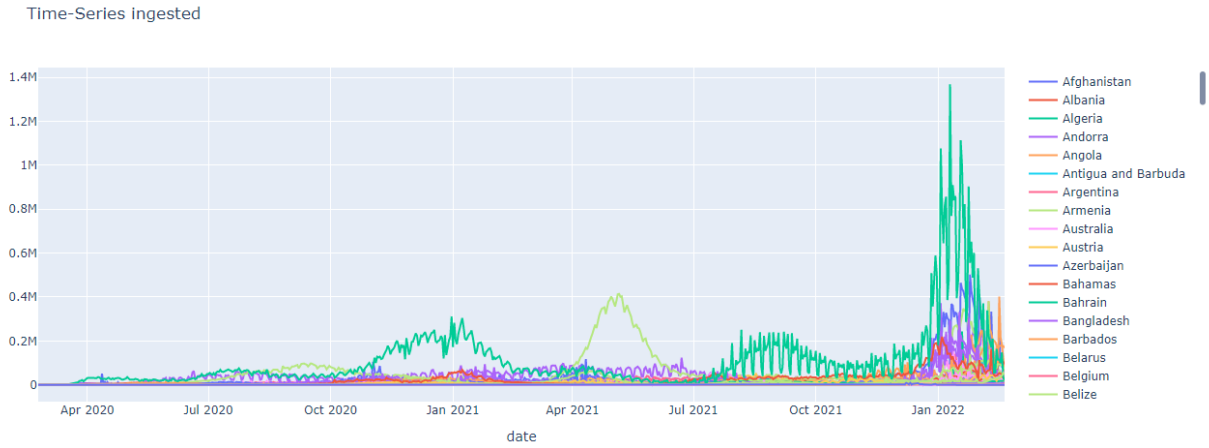


Figure 4.3: Time-series ingested from the COVID-19 dataset.

Finally, it can be appreciated in the Figure 4.3 the time-series ingested using the tool available in TIMEX-CLUSTERING; through this plot, it can be appreciated the time evolution of the data. By analyzing the dimensions of all the time-series it can be determined that for this dataset applying a pre-transformation to the data, as the logarithmic or modified logarithmic explained in subsection 3.3.2, will lead to improve the accuracy of the clustering algorithms. The reason of this is that, without normalizing data, the

series that looks like each other will be seen so different from each other and will affect the accuracy of the clustering process. This example of preprocessing the data will be seen with more detail in the following chapter.

4.2.1. Experiments' structure COVID-19 dataset

The structure of the experiments for this dataset will be the following: it will be applied the three clustering approaches, observation, feature and model based, to compare them with each other, analyzing advantages and drawbacks of each one of them. The clustering algorithms applied will be K-Means (observation and feature based approaches) and Gaussian Mixture model (model based approach). The distance metrics used will be Euclidean and DTW; soft-DTW will not be used due to the large volume of the dataset (the computational execution time was too long and the results obtained were similar to DTW). Results of the clustering with and without a data preprocessing transformation will be present, and for the Feature based approach the feature transformation used will be DWT using the Haar wavelet. It will be performed an exhaustive search over the specified parameter configurations and over a range from 3 to 6 number of clusters. Table 4.2 summarizes the major components used for the experiment on the COVID-19 dataset.

Clustering approach	Variable	PreTrans-formation	Distance metrics	Feature repre-sentation	Model	Evaluation criterion
Observation based	Univariate	None & Log-Modified	Euclidean DTW	N/A	K-Means	Silhouette score
Feature based	Univariate	None & Log-Modified	Euclidean DTW	DWT, Haar wavelet	K-Means	Silhouette score
Model based	Univariate	None & Log-Modified	Euclidean DTW	N/A	Gaussian Mixture Model	Silhouette score

Table 4.2: Structure of the experiments for the COVID-19 dataset.

4.3. Summary

Prior to move to the next chapter, it can be seen in the Table 4.3, as a reference, the datasets that were examined with the TIMEX-CLUSTERING framework.

Dataset	Ground truth classes	Number of time-series	Length, daily samples	Type	Source
Synthetic dataset: Trace	Known	50	40	Synthetic	UCR & UEA repository [4]
COVID-19 new positive cases	Unknown	181	727	Real case scenario	Our World in Data repository [38]

Table 4.3: List of datasets to assess.

In the next *Chapter 5*, the most relevant insights and lesson-learned provided by TIMEX-CLUSTERING on both datasets are described and commented; in particular, considerations on in the COVID-19 pandemic spread around the world up to February 2022.

5 | Experimental results

In this chapter, the results of the experiments performed while working on the thesis will be discussed. As mentioned in the subsections 4.1.1 and 4.2.1, all the different characteristics of the TIMEX-CLUSTERING framework, including all the clustering approaches, distance/similarity metrics, data transformations and clustering models will be tested; the goals of these experiments are to show the reliability of the framework against two completely different datasets, the efficiency and quality of the clustering obtained, in order to appreciate the TIMEX-CLUSTERING capabilities and to distinguish areas of future developments and improvements.

It is important to reiterate the main objective of this work, which is to develop an automatic clustering framework, which could be implemented for any user with a minimal amount of knowledge about clustering procedures, and to provide high efficiency and quality time-series clustering. Therefore, the most innovative features of TIMEX-CLUSTERING framework are:

- provide in an "as-a service" manner a fully automatic end-to-end clustering pipeline;
- deliver high efficiency and quality time-series clustering;
- offer an automatic framework where the three main clustering approaches can be compared with each other.

It is significant to mention that for both the datasets used, there are or could be algorithms developed in the clustering field with a greater efficiency; however, these works do not have two of the three features previously mentioned: provide to the final user an innovative framework, allows the user to use a fully automatic end-to-end clustering pipeline and allows the user to compare several clustering approaches with each other using a unique tool.

5.1. UCR & UEA Trace dataset results

In this section the steps followed to develop the experiments will be explained, and the most relevant results provided by the framework will be analyzed in detail for the UCR & UEA Trace dataset [4].

5.1.1. Data ingestion

For this step in the clustering process, as explained in the section 3.2, only two input files are needed:

- the dataset, which is a CSV file storing the M number of univariate time-series. This dataset is available as an online resource [45], the URL is specified in the JSON configuration file and the framework will download it automatically.
- a configuration file, named $param\_config$, specified in *Appendix A*, detailing the parameters that are available in the TIMEX-CLUSTERING.

Once that the mentioned files are prepared, the user only need to ingest the data by three simple lines of code on Python:

```
pip install timexseries_clustering
from timexseries_clustering.data_ingestion import ingest_timeseries
ingested_dataset = ingest_timeseries(param_config)
```

Listing 5.1: Python data ingestion.

This code, as explained in section 3.3, creates the data pre-processing of the dataset, with the goal of transforming the time-series present in the dataset into a form that the data clustering step can easily parse. In this stage three fully automatic different phases are involved: handling missing data, data pre-transformation and feature transformation. The output of these lines of code show to the user the relevant information of this data pre-processing stage:

```
INFO:timexseries_clustering.data_ingestion:Starting the data ingestion
    phase.
INFO:timexseries_clustering.data_ingestion:Finished the data-ingestion
    phase. Some stats:
-> Number of rows: 40
-> Number of columns: 50
-> Column names: ['timeseries1', 'timeseries2', ..., 'ts45', 'ts46', '
    ts47', 'ts48', 'ts49', 'ts50']
```

```
-> Number of missing data: [0, 0, ..., 0, 0, 0, 0, 0]
```

Listing 5.2: Output of data ingestion.

An example of the Pandas DataFrame provided by the data ingestion step is given in the Figure 5.1.

Date	timeseries1	timeseries2	timeseries3	timeseries4	timeseries5	ts6	ts7	ts8	ts9	ts10	...	ts41	ts42	ts43	ts44	ts45	ts46	ts47	ts48	ts49	ts50
2020-02-25	0.70	0.78	-1.45	0.75	-2.03	0.67	0.58	0.65	0.68	-1.35	...	0.55	0.63	0.62	0.54	0.75	0.61	-1.45	0.76	-1.38	0.67
2020-02-26	0.67	0.77	-1.48	0.76	-2.12	0.73	0.60	0.61	0.63	-1.37	...	0.54	0.60	0.63	0.58	0.75	0.63	-1.40	0.78	-1.36	0.57
2020-02-27	0.66	0.74	-1.50	0.78	-2.13	0.71	0.59	0.56	0.68	-1.39	...	0.59	0.61	0.63	0.59	0.75	0.67	-1.39	0.80	-1.37	0.65
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2020-04-02	0.58	0.79	0.73	0.65	0.53	0.63	0.58	0.59	0.61	0.78	...	0.58	0.55	0.50	0.60	0.65	0.61	0.76	0.75	0.77	0.56
2020-04-03	0.59	0.77	0.73	0.78	0.55	0.66	0.55	0.63	0.64	0.75	...	0.60	0.60	0.49	0.57	0.72	0.56	0.73	0.75	0.81	0.56
2020-04-04	0.60	0.79	0.71	0.73	0.51	0.68	0.59	0.63	0.65	0.72	...	0.58	0.61	0.57	0.64	0.66	0.56	0.75	0.75	0.83	0.56

40 rows × 50 columns

Figure 5.1: Pandas DataFrame provided by the ingestion of the Trace dataset.

5.1.2. Data clustering

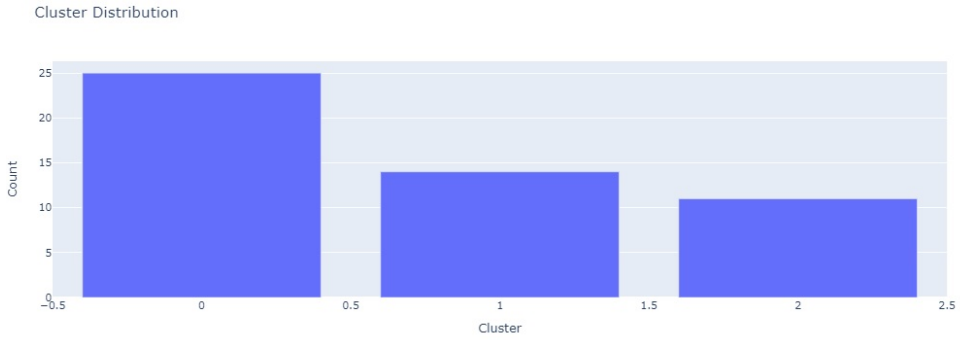
After the data pre-processing phase is finished, the user will obtain a Pandas DataFrame object, from now called *ingested_dataset*. The next step is to execute the fully automatic pipeline, where TIMEX-CLUSTERING computes the time-series clustering, according with all the parameters specified in the configuration file; the detail of these parameters can be appreciated in the Table 4.1 and in the configuration file in the *Appendix A*. To compute automatically the clustering pipeline the user only needs to execute two simple lines of code in Python:

```
from timexseries_clustering.data_clustering.pipeline import
    create_timeseries_containers
timeseries_containers = create_timeseries_containers(ingested_dataset,
    param_config)
```

Listing 5.3: Data clustering code.

In this stage, as explained in section 3.3, TIMEX-CLUSTERING automatically learns the univariate clustering models for each clustering approach indicated, and the output of this phase provides and highlights to the user relevant information: the best clustering model for each clustering approach, plots detailing the clusters distribution, subplots for each distance metric showing the cluster centers with the time-series that belongs to that cluster, a data frame indicating the cluster to which each time-series belong and parameters configuration that provides the largest evaluation metric score.

An example of the cluster distribution and the data frame indicating the cluster to which each time-series belongs, for the best clusters obtained in the observation based clustering approach, can be observed in the Figure 5.2.



Cluster 0	Cluster 1	Cluster 2
timeseries1	timeseries3	ts7
timeseries2	timeseries5	ts17
timeseries4	ts10	ts19
ts6	ts11	ts23
ts8	ts12	ts25
ts9	ts16	ts30
ts13	ts20	ts32
ts14	ts22	ts33
ts15	ts26	ts36
ts18	ts28	ts38
ts21	ts29	ts50
ts24	ts35	null

Figure 5.2: Data distribution and clustering results for the observation based approach, Trace dataset.

An important remark is that, while the previous lines of code are being executed, and the framework is computing the automatic clustering pipeline, the user receives as an output-feedback the state of the data clustering computation until the process finished:

```
INFO:Using approach: observation_based and using model k_means...
INFO:Computing univariate clustering using approach: observation_based,
    number of clusters: 3, distance metric: euclidean and transformation:
     none...
INFO:Computing univariate clustering using approach: observation_based,
    number of clusters: 4, distance metric: euclidean and transformation:
     none...
INFO:For the metric: euclidean the best clustering is obtained using 3
    number of clusters and transformation none.
INFO:For the model: k_means the best clustering is obtained using metric
     euclidean, with 3 number of clusters, and transformation none.
INFO:Process of observation_based clustering finished.
```

Listing 5.4: Output of data clustering.

5.1.3. Service delivery

With the results obtained in the previous section and by using the tools available in TIMEX-CLUSTERING, a website is delivered by means of the open-source Python library Dash3; an interactive web page highlighting the results obtained using this tools is created, simplifying their comprehensibility and simplifying their visualization in plots, tables and summaries. One of the main advantages of creating this website is its accessibility to a large audience. The TIMEX-CLUSTERING website created can be seen in the Figure 5.3. In the next section each result obtained is analyzed and discussed.

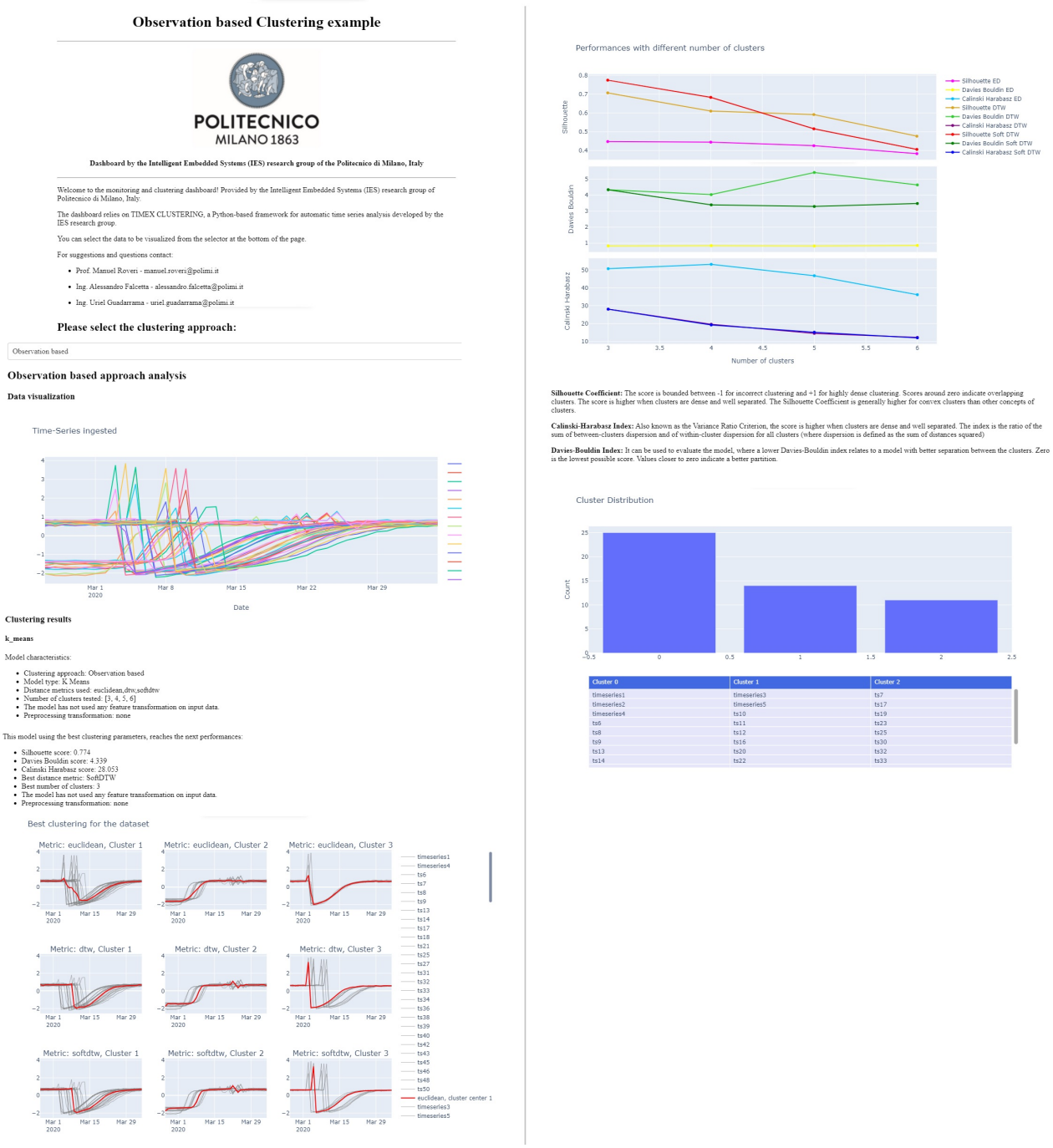


Figure 5.3: Website created based on the clustering's results of the Trace dataset.

The user can interact with the website, by first selecting one of the three clustering approaches that would like to analyze, among the three possible approaches: Observation, Feature or Model based approach.

Once selected the clustering approach, it is showed all the time-series ingested; with this graph as well as with all the graphs in the website, the user can interact with the data by

zooming in or zooming out and specific area or choosing a desired time interval, moreover through the mouse pointer the user can visualize specific information on the time-series, with the legend box the user can only visualize specific time-series, and many functions more thanks to the versatile Python library Dash3.

Later, the user can visualize essential information concerning each model used within the clustering approach selected, where the model's characteristics and the best clustering parameters achieved are described.

The best clustering parameters for each model is translated into a more comprehensible plot, where each row corresponds to the result of a different clustering. In a row, each subplot corresponds to a cluster. It represents the set of time-series from the dataset that were assigned to the considered cluster (in grey) as well as the cluster center of the cluster (in red). An example is shown in Figure 5.4.

The next graph is also a subplot, illustrating the performances for each number of clusters assessed for that clustering algorithm; for every evaluation criteria (Silhouette score, Davies Bouldin score or Calinski-Harabasz score) it is mapped in a subplot their corresponding score values. Finally, it can be appreciated the distribution of the time-series in the cluster distribution chart. Following this chart, a Table lists the set of time-series that were designated to every cluster, for a better comprehensibility of the clustering results to the user. The code used to create the interactive website can be found in the *Appendix A*.

5.1.4. Analysis of the results

According with the structure of the experiments explained in Section 4.1.1 and in the Table 4.1 for the UCR & UEA Trace dataset, the final results of these experiments are reported in the following Table 5.1. This table shows the best accuracy obtained by each clustering model on the dataset.

Clustering approach	Model	PreTrans-formation	Distance metrics	Feature repre-sentation	Best No. of clusters	Silhouette score
Observation based	K-Means	None	Euclidean	N/A	3	0.448
			DTW		3	0.707
			soft-DTW		3	0.774
		Log-Modified	Euclidean		3	0.538
			DTW		3	0.643
			soft-DTW		3	0.503
Feature based	K-Means	None	Euclidean	DWT	3	0.571
			DTW		4	0.573
			soft-DTW		4	0.619
		Log-Modified	Euclidean		3	0.589
			DTW		3	0.567
			soft-DTW		3	0.596
Model based	Gaussian Mixture Model	None	Log-Likelihood	N/A	3	0.216
		Log-Modified			3	0.242

Table 5.1: Structure of the experiments for the UCR & UEA Trace dataset.

To accomplish these results, it was performed an exhaustive search over the specified parameter configurations and over a range from 3 to 6 number of clusters.

Best clustering approach

According with the Table 5.1, the best-performing approach was the Observation based clustering approach, using a K-Means clustering model, applying no data pre-transformation and employing soft-DTW as distance metric; this model settles that the best number of clusters to group the data is 3 and presents a silhouette score of 0.774, which is close to 1 indicating that the clusters are dense and well separated.

On the other hand, the worst-performing approach was the Model based clustering, using Gaussian Mixture clustering models, applying no data pre-transformation and employing Log-Likelihood as distance metric; it presents a silhouette score of 0.216, which is closer to zero indicating an overlapping of the clusters.

By observing the comparison of the best performing model against the worst performing

model, in the Figure 5.4 and Figure 5.5 respectively, it can be appreciated a much better clustering quality in the best performing model, because this model could better cluster the classes beforehand known and explained in Section 4.1. Indeed, it was described the behaviour of each class; the first class (*Cluster*1 graph in the left side of the Figure 5.4) has a behavior that can be distinguished from the third class (*Cluster*3 graph in the right side of the Figure 5.4) by the presence of a high frequency peak at the beginning of the transient; the second class (*Cluster*2 graph in the middle of the Figure 5.4) has a different behavior with a small burst in the middle phase of the transient. In comparison, the worst-performing model, Figure 5.5, could not distinguish properly the behavior of these three clusters, result that can be corroborated with the silhouette score 0.774 in the best-performing model against a score of 0.216 in the worst-performing model, indicating that the final clusters results are dense and accurately separated.

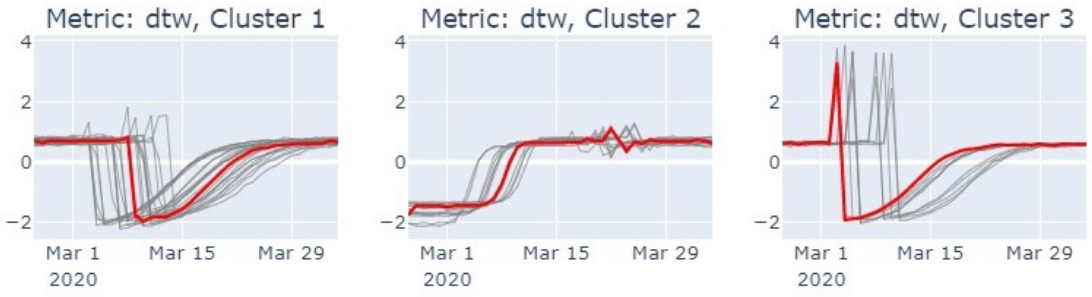


Figure 5.4: Best-performing result, Observation based clustering approach, K-Means clustering model, no data pre-transformation and employing soft-DTW as distance metric.

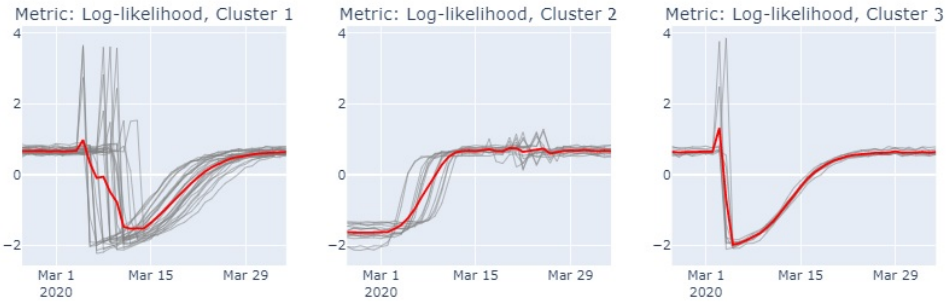
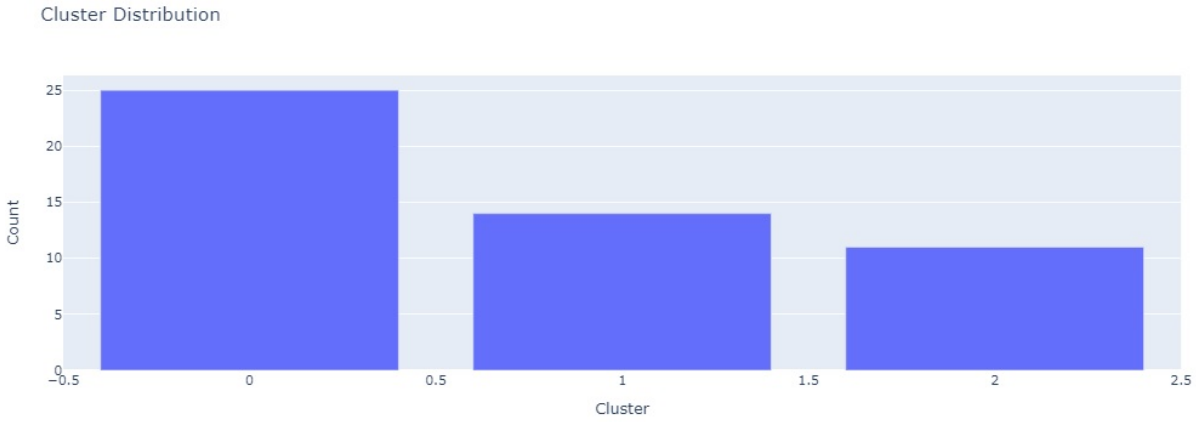


Figure 5.5: Worst-performing result, Model based clustering approach, Gaussian Mixture models, no data pre-transformation and employing Log-Likelihood as distance metric.

Finally, the quality of the best-performing model results can be corroborated with the cluster distribution plot, Figure 5.6, where it can be observed that the time-series are equally distributed to clusters. In the case that this graph would show that the time-series

are not equally distributed, that would be an indicator of a not optimal clustering model, due to many possible factors as: curse of dimensionality, ill-posed clustering approach or ill-posed model parameter for the dataset, etc. An example of this case scenario can be seen in the results of the COVID-19 dataset, subsection 5.2.4; where a discussion on how the problem was approached and solved is presented.



Cluster 0	Cluster 1	Cluster 2
timeseries1	timeseries3	ts7
timeseries2	timeseries5	ts17
timeseries4	ts10	ts19
ts6	ts11	ts23
ts8	ts12	ts25
ts9	ts16	ts30
ts13	ts20	ts32
ts14	ts22	ts33
ts15	ts26	ts36
ts18	ts28	ts38
ts21	ts29	ts50
ts24	ts35	null

Figure 5.6: Best-performing cluster distribution, with observation based approach, K-means model, no data pre-transformation and soft-DTW distance metric.

In the Figure 5.7, the accuracy for each model for the Observation based approach can be observed; these were obtained with the exhaustive search over the specified parameter configurations and over the range from 3 to 6 number of clusters, indicating that the best number of cluster for the Trace dataset was 3, obtained with the best-performing model already analyzed.

Figure 5.7: Performance with different number of clusters, Observation based approach, K-means model, no data pre-transformation and soft-DTW distance metric.

Comparison of clustering approaches performances

Let us recall the goal of using this synthetic dataset: to validate the correct clustering operation by knowing beforehand the ground truth classes. It can be concluded that the three clustering approaches and the TIMEX-CLUSTERING framework functioned satisfactorily and accomplished this goal; indeed, through the three different approaches, it was validated that our automatic pipeline could achieve the same three clusters, and it was able to cluster the behaviors already expected and explained in the Section 4.1. This experiment proved that the three approaches were effective for tackling the task at hand.

Furthermore, when comparing the three clustering approaches performances, it was validated the hypothesis stated also in the Section 4.1, which expressed that the Observation based approach would reach a higher quality clustering for this dataset, obtaining the best silhouette score of 0.774. This was because as explained, the Observation based approach is more suitable for datasets that have short length time-series, as the case of the UCR & UEA Trace dataset.

In comparison, the Model and Feature based clustering approaches achieved their best silhouette score of 0.242 and 0.619 respectively. The lower accuracy of the Model based approach can be attributed to the assumption that the dataset is generated from a mixture of a finite number of Gaussian distributions, assumptions that might be incorrect specifically for this dataset; besides, this approach reduces its performance when clusters are close to each other, as in this case scenario where two clusters have a similar behavior only distinguished one from another by the presence of a high frequency peak at the beginning of the transient. Regarding the accuracy of the Feature based clustering of 0.619, the obtained results are acceptable despite the short length of the time-series. However, this could be improved by choosing another feature transformation method to extract better

features, because through the DWT transformation an important dimensional reduction is attained and consequently computational time saving is achieved.

Concerning the computational time execution of each clustering approach, in the Table 5.2 can be found the execution time for each clustering approach. These executions were performed in a serverless Jupyter notebook environment for interactive development provided by Google Colaboratory (more commonly referre as Google Colab [13]). It is important to mention that a Graphic processing unit (GPU) was used as a tool to accelerated the clustering computation (the GPU was also provided by the Google Colab environment).

Clustering approach	Model	PreTrans-formation tested	Distance metrics tested	Feature repre-sentation	No. of clusters tested	Time execution sec
Observation based	K-Means	None & Log-Modified	Euclidean DTW soft-DTW	N/A	3, 4, 5, 6	77
Feature based	K-Means	None & Log-Modified	Euclidean DTW soft-DTW	DWT	3, 4, 5, 6	32
Model based	Gaussian Mixture Model	None & Log-Modified	Log-Likelihood	N/A	3, 4, 5, 6	2

Table 5.2: Computational time execution of each clustering approach, Trace dataset.

From the results shown in the Table 5.2, it can be highlighted that the Feature based approach was executed in half the time of the Observation based approach (77 seconds), representing a significant computational time saving thanks to dimensional reduction of the dataset, as expected and described in previous chapters. The Model based approach was the approach executed in the less amount of time, due to the assumption that the dataset is generated from a specific kind of model, in this case Gaussian distributions, and also because in the other two approaches more parameter combinations were tested; in general K-Means requires much less time than the required by GMM [33].

Finally, two aspects are commented. The first one is that as it can be perceived in the Table 5.1, the results obtained without a data pre-transformation were higher than the results obtained using a data pre-transformation. This is since, as explained in Section 5.1, for this dataset there was no need to apply a pre-transformation, because the time-series

scales were not so different between each other (the opposite situation happens for the COVID-19 dataset, which will be explained with detail in the subsection 5.2.4). The second and more interesting comment is that, for this dataset, in the Observation and Feature based approaches, the distance metric performed a key aspect to achieve the most efficient and high quality clustering; in the Figure 5.5, it can be appreciated (in the graph on the left) that by using Euclidean metric, the algorithm could not discriminate correctly some time-series that presented the high frequency peak at the beginning of the transient; therefore, the algorithm assigned those time-series to an incorrect cluster. On the other hand, in the Figure 5.4, by using DTW or soft-DTW as distance metric, the algorithm could cluster correctly the three different behaviors, since these metrics help to identify time-series with similar shape and different phase, as in our dataset were the time-series behaviors are diphased; in conclusion the most suitable distance metric were DTW and soft-DTW, results that can be corroborated in the Table 5.1, were the higher silhouette scores were the ones employing DTW or soft-DTW as distance measurement.

5.2. COVID-19 dataset results

In this section, the steps followed to develop the experiments for the COVID-19 pandemic case will be explained, the most relevant results provided by the framework will be analyzed in detail.

5.2.1. Data ingestion

For this step in the clustering process, as explained in the section 3.2, only two input files are needed:

- the dataset, which is a CSV file storing the M number of univariate time-series. This dataset is available as an online resource [46], the URL is specified in the JSON configuration file and the framework will download it automatically.
- a configuration file, named $param\_config\_covid$, specified in *Appendix A*, detailing the parameters that are available in the TIMEX-CLUSTERING.

Once that the mentioned files are prepared, the user only needs to ingest the data by three simple lines of code in Python:

```
pip install timexseries_clustering
from timexseries_clustering.data_ingestion import ingest_timeseries
```

```
ingested_dataset_covid = ingest_timeseries(param_config_covid)
```

Listing 5.5: Python data ingestion.

This code, as explained in section 3.3, creates the data pre-processing of the dataset, with the goal of transforming the time-series present in the dataset into a form that data clustering step can easily parse. In this stage three fully automatic different phases are involved: handling missing data, data pre-transformation and feature transformation. The output of these lines of code show to the user the relevant information of this data pre-processing stage:

```
INFO:timexseries_clustering.data_ingestion:Starting the data ingestion
    phase.
INFO:timexseries_clustering.data_ingestion:Finished the data-ingestion
    phase. Some stats:
-> Number of rows: 728
-> Number of columns: 181
-> Column names: ['Afghanistan', 'Albania', 'Algeria', ..., 'United
    States', 'Uruguay', 'Venezuela', 'Vietnam', 'Yemen', 'Zimbabwe']
-> Number of missing data: [0, 0, 0,..., 0, 0, 0, 0, 0, 0]
```

Listing 5.6: Output of data ingestion.

An example of the Pandas DataFrame provided by the data ingestion step is given in the Figure 5.8.

	Afghanistan	Albania	Algeria	Andorra	Angola	Antigua and Barbuda	Argentina	Armenia	Australia	Austria	...	United Kingdom	United States	Uruguay	Uzbekistan	Vatican	Venezuela	Vietnam	Yemen	Zambia
date																				
2020-02-24	5	0	0	0	0	0	0	0	0	0	...	2	0	0	0	0	0	0	0	0
2020-02-25	0	0	1	0	0	0	0	0	0	2	...	4	0	0	0	0	0	0	0	0
2020-02-26	0	0	0	0	0	0	0	0	0	0	...	3	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2022-02-18	184	300	316	67	20	5	15389	1417	23217	31166	...	47481	147827	7179	272	0	979	42439	13	290
2022-02-19	74	303	251	64	12	8	7807	1469	17376	25147	...	34011	33556	4355	191	0	606	54830	10	252
2022-02-20	274	260	118	64	21	11	4450	700	18219	48357	...	25536	22053	3337	204	0	833	47200	5	178

728 rows × 181 columns

Figure 5.8: Pandas DataFrame provided by the ingestion of the COVID-19 dataset.

5.2.2. Data clustering

After the data pre-processing phase is finished, the user will obtain a Pandas DataFrame object, from now called *ingested_dataset_covid*. The next step is to execute the fully

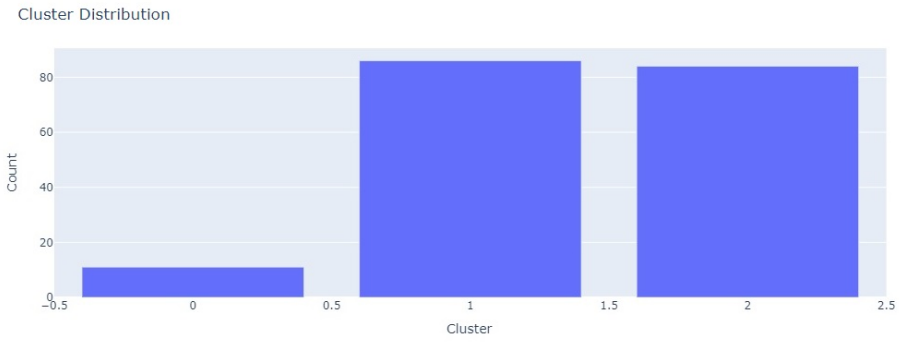
automatic pipeline, where TIMEX-CLUSTERING computes the time-series clustering, according with all the parameters specified in the configuration file (the detail of these parameters can be appreciated in the Table 4.1 and in the configuration file in the *Appendix A*). To compute automatically the clustering pipeline the user only needs to execute two simple lines of code in Python:

```
from timexseries_clustering.data_clustering.pipeline import
    create_timeseries_containers
timeseries_containers = create_timeseries_containers(
    ingested_dataset_covid, param_config_covid)
```

Listing 5.7: Python data ingestion.

This stage, as explained in section 3.3, for the dataset ingested TIMEX-CLUSTERING automatically learns the univariate clustering models for each clustering approach indicated, and the output of this phase provides and highlights to the user relevant information: the best clustering model for each clustering approach, plots detailing the clusters distribution, subplots for each distance metric showing the cluster centers with the time-series that belongs to that cluster, a data frame indicating the cluster to which each time-series belong and parameters configuration that provides the largest evaluation metric score.

An example of the cluster distribution and the data frame indicating the cluster to which each time-series belongs, for the best clusters obtained in the feature based clustering approach, can be observed in the Figure 5.9.



Cluster 0	Cluster 1	Cluster 2
Argentina	Afghanistan	Albania
Brazil	Andorra	Algeria
France	Angola	Armenia
Germany	Antigua and Barbuda	Australia
India	Bahamas	Austria
Italy	Barbados	Azerbaijan
Russia	Belize	Bahrain
Spain	Benin	Bangladesh
Turkey	Bhutan	Belarus
United Kingdom	Brunei	Belgium
United States	Burkina Faso	Bolivia
null	Burundi	Bosnia and Herzegovina

Figure 5.9: Data distribution and clustering results for the observation based approach, COVID-19 dataset.

5.2.3. Service delivery

With the results obtained in the previous section and by using the tools available in TIMEX-CLUSTERING, a website is delivered by means of the open-source Python library Dash3; it was employed the same structure and procedure followed for the UCR & UEA Trace dataset, subsection 5.1.3.

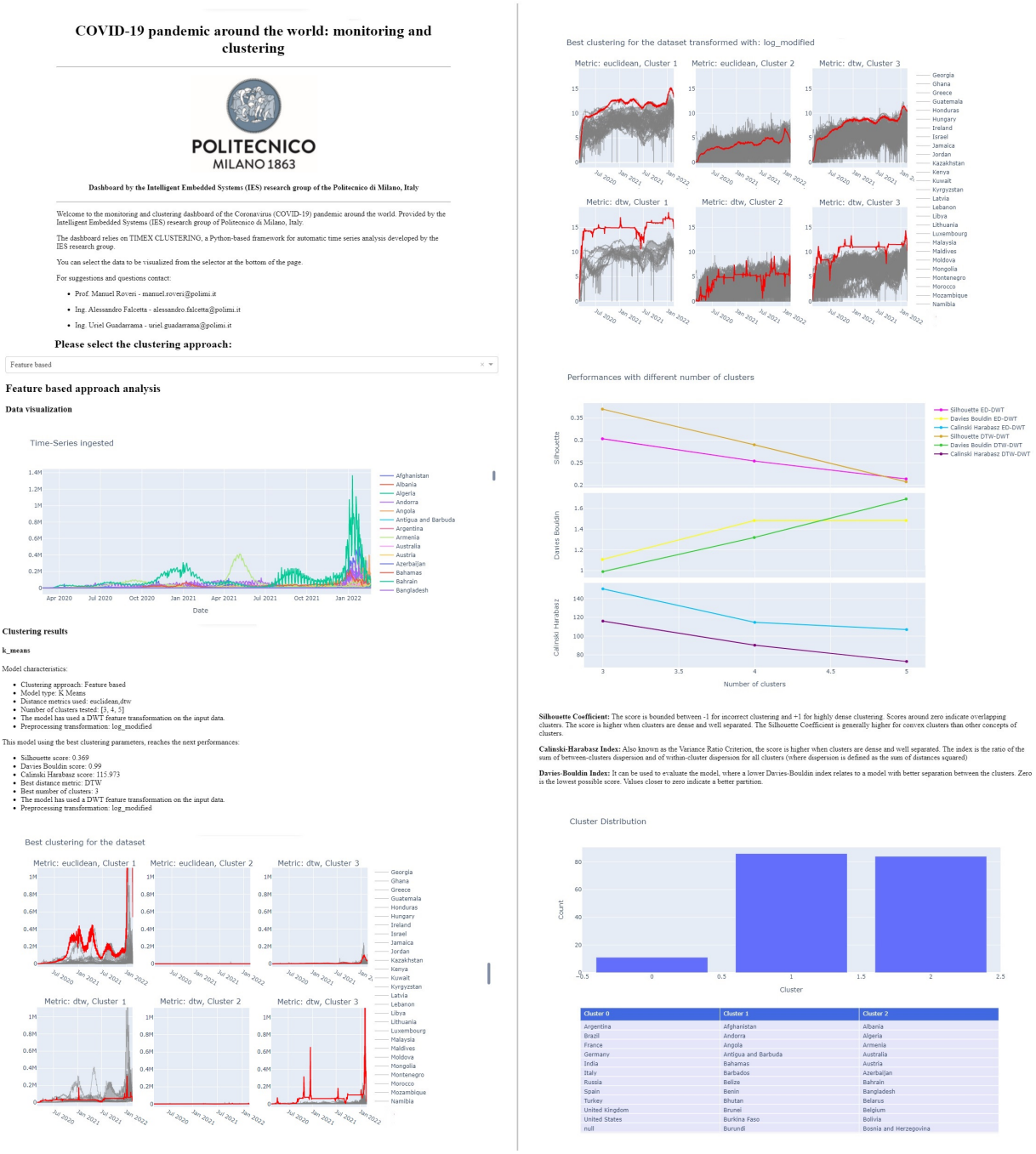


Figure 5.10: Website created based on the clustering's results of the COVID-19 dataset.

5.2.4. Analysis of the results

According with the structure of the experiments explained in Section 4.2.1 and in the Table 4.2 for the COVID-19 dataset, the final results of these experiments are reported in the following Table 5.3. This table shows the best accuracy obtained by each clustering model on the dataset. To accomplish these results, it was performed an exhaustive search

over the specified parameter configurations and over a range from 3 to 6 number of clusters.

Clustering approach	Model	PreTrans-formation	Distance metrics	Feature repre-sentation	Best No. of clusters	Silhouette score
Observation based	K-Means	None	Euclidean	N/A	3	0.908
			DTW		3	0.870
		Log-Modified	Euclidean		3	0.295
			DTW		3	0.332
Feature based	K-Means	None	Euclidean	DWT	3	0.910
			DTW		3	0.862
		Log-Modified	Euclidean		3	0.303
			DTW		3	0.369
Model based	Gaussian Mixture Model	None	Log-Likelihood	N/A	3	0.886
		Log-Modified			3	0.279

Table 5.3: Structure of the experiments for the COVID-19 dataset.

Best clustering approach

According only with the Table 5.3, the best-performing approach would be the Feature based clustering approach, using a K-Means clustering model, applying no data pre-transformation and employing Euclidean as distance metric; this model settles that the best number of clusters to group the data is 3 and presents a silhouette score of 0.910, which is close to 1 indicating that the clusters are dense and well separated. However, by analyzing the Figure 5.11 and the Figure 5.12, it can be seen that the cluster quality achieved from this model is not acceptable even though the silhouette score is high. This is because the Figure 5.11 shows that the model clustered independently United States and India in one cluster for each of them, and the rest of the countries in the remaining cluster. This is confirmed in the Figure 5.12 that shows the cluster distribution of the countries. The time-series are not equally distributed to clusters due to a lack of data pre-transformation, which is needed because of the different scales in the COVID-19 time-series. This transformation would have made the data features more significant for the clustering steps. This issue also appeared for the Observation and Model based clustering approaches with no data pre-transformation.

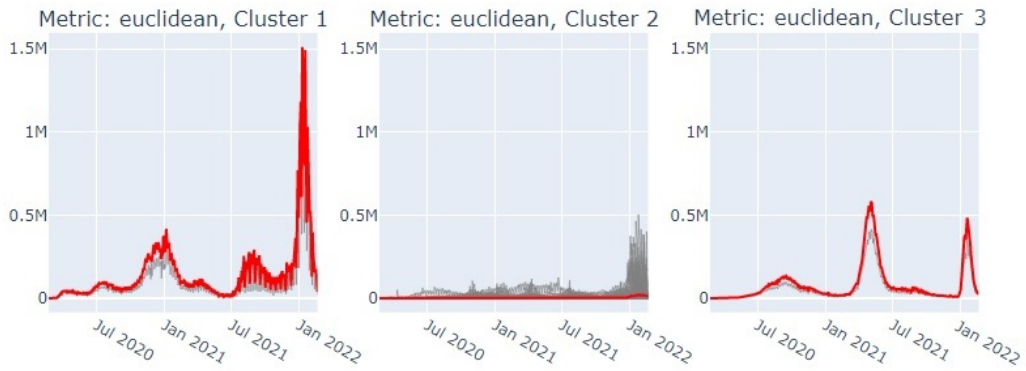


Figure 5.11: Worst-performing result, Feature based clustering approach, K-Means, no data pre-transformation, no feature transformation and Euclidean distance metric.

Figure 5.12: Worst-performing cluster distribution, with Feature based approach, K-means model, no data pre-transformation, no feature transformation and Euclidean distance metric.

Looking closely at the Figure 5.13, the first cluster (*Cluster*1 graphs in the left side of the Figure 5.13) grouped the countries that faced different infection peaks through all the pandemic. In this cluster 11 countries can be found: Italy, France, Spain, United States, India, Germany, India, Russia, Turkey, United Kingdom, Argentina and Brazil. The second class (*Cluster*2 graphs in the middle of the Figure 5.13) grouped the countries that better controlled the pandemic: in this cluster 86 countries can be found, examples of these countries are: China, New Zealand, Montenegro, Hong Kong, Congo, Iceland,

so on and so forth. The third class (*Cluster*3 graphs the right side of the Figure 5.13) grouped the countries that also faced different infection peaks through all the pandemic, but that did not have an exponential growing in the new positive cases. Portugal for example, faced the biggest infection peak with the Omicron variant of the virus; however, in comparison with France (that belongs to the cluster 1), the new positive cases did not grow exponentially, sign that Portugal controlled to a greater extent this last peak of infections. In this last cluster 84 countries can be found, examples of these countries are: Portugal, Australia, Belgium, Egypt, Mexico, South Korea and many more; in this cluster also countries that controlled almost totally the positive cases through all the pandemic, but that were affected with an increase in the number of infections provoked by the Omicron variant of the virus can be found (e.g. South Korea).

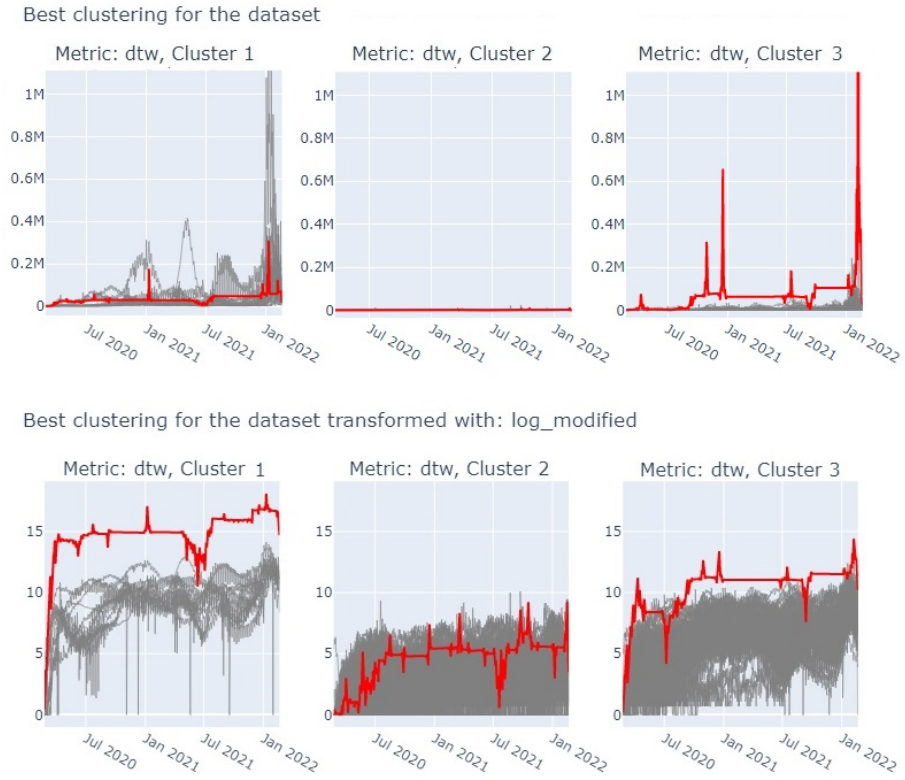
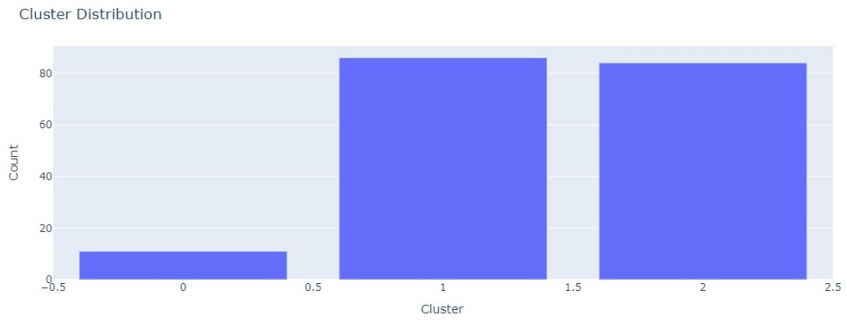


Figure 5.13: Best-performing result, Feature based approach, K-Means, Log-Modified data pre-transformation, DWT feature transformation and DTW distance metric.



Cluster 0	Cluster 1	Cluster 2
Argentina	Afghanistan	Albania
Brazil	Andorra	Algeria
France	Angola	Armenia
Germany	Antigua and Barbuda	Australia
India	Bahamas	Austria
Italy	Barbados	Azerbaijan
Russia	Belize	Bahrain
Spain	Benin	Bangladesh
Turkey	Bhutan	Belarus
United Kingdom	Brunei	Belgium
United States	Burkina Faso	Bolivia
null	Burundi	Bosnia and Herzegovina

Figure 5.14: Best-performing cluster distribution, Feature based approach, K-means model, Log-Modified data pre-transformation, DWT feature transformation and DTW distance metric.

Finally, the quality of the best-performing model also can be corroborated with the cluster distribution plot, shown in Figure 5.14, where it can be observed that the time-series are equally distributed to clusters. In the Figure 5.15, there are the accuracy for each model, for the Feature based approach, over a range from 3 to 6 number of clusters, indicating that the best number of cluster for the COVID-19 dataset was 3, obtained with the best-performing model already analyzed.

Figure 5.15: Performance with different number of clusters, Feature based approach, K-means model, Log-Modified data pre-transformation, DWT feature transformation and DTW distance metric.

Comparison of clustering approaches performances

From the Table 5.3 and the previous subsection 5.2.4, it can be highlighted that the best clustering results were obtained employing a data pre-transformation (the detailed explanation of this is discussed in the subsection 5.2.4). Moreover, it can also be highlighted that, for Observation and Feature based approaches, the best clustering results were obtained employing DTW as distance metric, due to the nature of the COVID-19 virus propagation. Indeed, the peaks of new positive cases in each country present time shifts among the countries; for this reason the most suitable distance metric was the DTW, aspect that was validated by the results shown in the Table 5.3, where the higher silhouette scores were the ones employing DTW as distance measurement and Log-Modified as data pre-transformation.

Furthermore, when comparing the three clustering approaches performances, it was validated the hypothesis stated also in the Section 4.2, which expressed that for this dataset it was expected to obtain clusters more efficiently and with higher quality by using the Model or Feature based clustering approaches. The best clustering model was effectively obtained through the Feature based approach, as explained with detail in the previous subsection 5.2.4, and also based on the efficiency (computational time saving) showed in the Table 5.4.

Concerning the computational time execution of each clustering approach, in the Table 5.4 can be found the execution time for each clustering approach. These executions were performed in a serverless Jupyter notebook environment for interactive development provided by Google Colaboratory (more commonly referredas Google Colab [13]). It is important to mention that a Graphic processing unit (GPU) was used as a tool to accelerated the clustering computation (the GPU was also provided by the Google Colab environment).

Clustering approach	Model	PreTrans-formation tested	Distance metrics tested	Feature repre-sentation	No. of clusters tested	Time execution min
Observation based	K-Means	Log-Modified	Euclidean DTW	N/A	3, 4, 5	80
Feature based	K-Means	Log-Modified	Euclidean DTW	DWT	3, 4, 5	30
Model based	Gaussian Mixture Model	Log-Modified	Log-Likelihood	N/A	3, 4, 5	22

Table 5.4: Computational time execution of each clustering approach, COVID-19 dataset.

From the results shown in the Table 5.4, it can be highlighted that the Feature based approach was executed in less than half the time of the Observation based approach (80 minutes), representing a significant computational time saving thanks to dimensional reduction of the dataset; the computational time saved is more notorious for this kind of datasets with longer length and also with many time-series within the dataset. This aspect, and the fact that the Model based approach obtained the best clustering results, confirms that Model based clustering approach is the most suitable one for this dataset. The Model based approach was the approach executed in the less amount of time, due to the assumption that the dataset is generated from a specific kind of model, in this case Gaussian distributions, and also because in the other two approaches more parameter combinations were tested; in general K-Means requires much less time than the one required by GMM [33].

The clustering results obtained with the Model and Observation based clustering approaches, with a silhouette score of 0.279 and 0.369 respectively, represent decent quality cluster results similar to the results obtained with the Feature based approach, with a silhouette score of 0.332. The quality of these results were also validated by looking at the cluster distribution of each model, as in the case of the Feature based approach explained in the subsection 5.2.4. However, the efficiency (computational time execution) of the Observation based clustering approach was not acceptable, because it was more than the double of time taken by the Feature based approach, aspect that might be crucial for a non-expert user without the knowledge of how to manipulate a GPU. Regarding the efficiency (computational time execution) of the Model based approach, which was of 22 minutes, we can say that this approach is the second most suitable clustering approach

for the COVID-19 dataset.

6 | Conclusions and future developments

The aim of this thesis was to introduce TIMEX-CLUSTERING, an open-source framework for clustering-as-a-service, that could be able to provide users an automatic tool to perform, with high efficiency and quality, time-series clustering. This framework can be implemented for any user with a minimal amount of knowledge about clustering procedures and can provide the template sets as accurate as those created by other clustering algorithms.

Mentioned the main objectives of this work, the most innovative characteristics of TIMEX-CLUSTERING framework are:

- provide in an "as-a service" manner a fully automatic end-to-end clustering pipeline, that could be employed for any user;
- provide high efficiency and quality time-series clustering;
- provide an automatic framework where the three main clustering approaches, Observation, Feature and Model based, can be compared with each other.

Based on the topics addressed so far in this work and based on our experiments, it will be determined if our objectives were satisfactorily fulfilled. In the *Chapter 3*, we outlined the structure of the TIMEX-CLUSTERING framework explaining every detail and foundation of the framework, which is capable of automatically cluster time-series. Then, in *Chapter 4*, we introduced the two datasets, the UCR & UEA Trace dataset and the COVID-19 pandemic new cases spread worldwide, that would test all the different characteristics of the TIMEX-CLUSTERING framework, including all the clustering approaches, distance/similarity metrics, data transformations and clustering models; the goal of these experiments was to show the reliability, efficiency and quality of the clustering of the framework against two completely different datasets.

After that, *Chapter 5* describes the experiments that were performed and shows the results obtained. Based on these results obtained, it can be concluded that TIMEX-

CLUSTERING was successfully applied to the clustering of the COVID-19 pandemic spread around the world and to the UCR & UEA Trace dataset.

The results obtained for the UCR & UEA Trace synthetic dataset, validated the reason behind using this dataset, which was to confirm the correct clustering operation by knowing beforehand the ground truth classes. So, after performing the experiments defined in Section 4.1, it can be concluded that the three clustering approaches used by TIMEX-CLUSTERING functioned satisfactorily and accomplished this goal. This is because, through the three different approaches, it was validated that our automatic pipeline could achieve the same three clusters already know beforehand, and TIMEX-CLUSTERING was able to cluster the behaviors already expected and explained in the Section 4.1.

Moreover, as proposed in the hypothesis stated in the Section 4.1, the Observation based approach was expected to reach the highest quality clustering for this dataset. Indeed, it obtained the best silhouette score of 0.774, because, as explained in Section 4.1, the Observation based approach was the most suitable approach this kind of dataset that have short length time-series. Through the best model obtained using Observation based approach, the clustering quality of the framework was validated, since TIMEX-CLUSTERING was able to cluster the same classes known in advance and described in the Section 4.1.

Regarding the results obtained for the COVID-19 dataset, the objectives behind using this dataset were to have the possibility to apply our framework in a real-case scenario, explore the evolution and statistics on the coronavirus pandemic around the world to identify the most significant homogeneous groups of countries and get a more comprehensive understanding of how the pandemic was managed in each country. This, in general, could lead to discover patterns and identify, for example, common measures taken by the countries that were most successful in the control of the virus propagation.

After performing the experiments defined in the Section 5.2, it can be concluded that these objectives were successfully fulfilled by TIMEX-CLUSTERING, because, through the best model obtained using Feature based approach, TIMEX-CLUSTERING was able to cluster the countries in three significant clusters. As explained with detail in subsection 5.2.4, the first cluster grouped the countries that faced different infection peaks through all the pandemic; in this cluster 11 countries could be found: Italy, France, Spain, United States, India, Germany, India, Russia, Turkey, United Kingdom, Argentina and Brazil. The second cluster grouped the countries that better controlled the pandemic; in this cluster 86 countries could be found, examples of these countries are: China, New Zealand, Montenegro, Hong Kong, Congo, Iceland, etc. The third cluster congregated the countries that also faced different contagion peaks through all the pandemic, but that did not have

an exponential growing in the new positive cases. Portugal, for example, faced the biggest contagion peak with the Omicron variant of the virus; however, in comparison with France (that belongs to the cluster 1), the new positive cases did not grow exponentially, sign that Portugal controlled to a greater extent this last peak of contagions. In this last cluster 84 countries could be found, examples of these countries are: Portugal, Australia, Belgium, Egypt, Mexico, South Korea and many more; in this last cluster there are also countries, e.g. South Korea, that controlled almost totally the positive cases through all the pandemic, but that were affected ultimately with an increase in the number of infections provoked by the Omicron variant of the virus.

Concerning the time-series clustering efficiency accomplished by TIMEX-CLUSTERING, the computational time execution of each clustering approach was shown with detail in the Table 5.2 for the UCR & UEA dataset and in the Table 5.4 for the COVID-19 dataset; we highlight that, for the first experiment, the three clustering approaches achieved a high efficiency computation by executing the clustering pipeline in less than two minute.

For the second experiment, the computational time execution for each clustering approach in the COVID-19 dataset lead to more complex conclusions, because the significant computational time saving achieve by the Feature based approach, manifests the benefits of TIMEX-CLUSTERING of having three different clustering approaches within its features, this characteristic benefits the range of different possible dataset that the user can cover, for which in some cases the best strategy would be to employ one clustering approach instead of other, but in all the cases the user will have a powerful tool and the sustenance of TIMEX-CLUSTERING to achieve high efficiency and quality time-series clustering as-a-service.

In general, it can be concluded that TIMEX-CLUSTERING represents a really useful tool for both data scientists and non-expert users, because it supports them in the process of creating and evaluating clustering models, it provides different clustering approaches, and it provides, in an as-a service manner, a fully automatic end-to-end clustering pipeline. We believe that further research and development of TIMEX-CLUSTERING, could lead in the near future to the creation of the most robust, complete and automated clustering as-a-service pipeline. Some future developments that would contribute and add great value to the framework are detailed in the following section.

6.1. Future developments

The growth in interest in time-series clustering has resulted in the development of a wide variety of techniques. However, as shown in the part of the related time-series litera-

ture, traditional time-series clustering tends to be based on non-automatic approaches; therefore, we believe that the fully automatic end-to-end pipeline developed during this thesis can be improved through deeper investigations and improvements of the TIMEX-CLUSTERING framework.

In the following list, some of the future works we consider more important to expand the TIMEX-CLUSTERING capabilities are outlined:

- **Multivariate Time-Series Clustering**: this future work would be the most challenging and probably the most enriching to the TIMEX-CLUSTERING framework, through an option that could allow the user to cluster multivariate time-series. It would enable the framework to discover more different types of behaviors that are manifested thanks to the influence of more variables. The clusters obtained through multivariate clustering could be used for tasks such as anomaly detection or systems maintenance. In our work, we proposed only univariate time-series clustering given our time constraints, indeed, implementing multivariate clustering would require a completely redefinition of each step in the pipeline (data ingestion, data preprocessing, data description, data clustering and service delivery).
- **Increasing clustering models**: for each clustering approach, additional clustering models would contribute to cover more datasets with different nature, as well as it would improve the efficiency and quality of the time-series clustering. For the Observation-based and Feature-based approaches, the first model to be implemented could be the agglomerative model for its visualization power in time-series clustering and because it is a hierarchical method that does not require the number of clusters as initial parameter; other methods that could be developed for these approaches are K-Medoids and Fuzzy K-Means, that could provide a point of comparison with respect to the already implemented K-Means method. For the Model-based approach it would be enriching to add more polynomial models (ARIMA, HMM, etc.) or add Neural Network methods, i.e. Self-Organization Map (SOM).
- **Adding more distance/similarity measures**: the choice of a proper distance metric depends on the characteristic of the time-series, length, representation method and on the objective of the clustering; for that reason, adding more distance metrics would help to cover more characteristics of datasets. For example, adding the Longest Common Subsequence (LCSS) distance metric would improve the discover of similar time-series in shape; or, in case of addition of an agglomerative clustering method, the Kullback-Leibler would be the most useful distance measurement.
- **Extending the representation methods**: adding more representation methods

would give to the user more options to reduce the dimensionality of the time-series, while still preserving the fundamental characteristics of the dataset; adding techniques such as Discrete Fourier Transformation (DFT) would add the option to describe in the frequency domain the dynamics of stationary time-series. Other kind of techniques, such as Single Value Decomposition (SVD) or Piecewise Aggregate Approximation (PAA) would also enrich the framework. It is important to mention that the DWT could be also improved itself, by allowing the user to choose its parameters such as type of wavelet, coefficient to use, etc.

- **Introducing external indexes evaluation criteria**: adding the option to the user to input, during the data ingestion phase, the ground truth classes, and employ an external index clustering evaluation method such as Rand Index (RI), Adjusted Rand Index, Entropy, Purity, Jacard, F-measure, Folkes and Mallow index (FM), Cluster Similarity Measure (CSM) or MNI. Even though, usually, it is not directly applicable in real-life unsupervised tasks, this kind of evaluation criteria could help to achieve a more accurate clustering.

- **Developing a new clustering approach branch**: create a new clustering approach that makes use of a hybrid method for clustering the time-series. There exits many studies to improve the quality of the representation methods and distance measurements; however, the hybrid models are a growing area in the field of time-series clustering. For that reason to increase the tools of our framework, it could be implemented a new hybrid clustering model or use an existing hybrid or multi-step model.

- **Testing new datasets**: test new datasets from the UCR & UEA repository [4], a highly recognized repository of the most influential research surrounding the subject of time-series classification and clustering, with the objective of compare the TIMEX-CLUSTERING framework performance against other optimized models. This could suggest different improvement areas in the framework, and it would validate its efficiency.

Bibliography

[1] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah. Time-series clustering - a decade review. *Information Systems*, 53:16–38, 2015. ISSN 0306-4379. doi: https://doi.org/10.1016/j.is.2015.04.007.

[2] F. Alessandro and R. Manuel. Timex: A framework for time-series forecasting-as-a-service. *Singapore, SG*, 2021.

[3] B. Andreopoulos, A. An, X. Wang, and M. Schroeder. A roadmap of clustering algorithms: Finding a match for a biomedical application. *Briefings in bioinformatics*, 10:297–314, 03 2009. doi: 10.1093/bib/bbn058.

[4] A. Bagnall, J. Lines, W. Vickers, and E. Keogh. The uea & ucr time series classification repository. *URL http://www. timeseriesclassification. com*, 122, 2018.

[5] R. Baragona. A simulation study on clustering time series with metaheuristic methods. *Quaderni di Statistica*, 3:1–26, 2001.

[6] J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[7] J. Beran and G. Mazzola. Visualizing the relationship between two time series by hierarchical smoothing models. *Journal of Computational and Graphical Statistics*, 8(2):213–238, 1999.

[8] M. Chiş, S. Banerjee, and A. E. Hassanien. Clustering time series data: an evolutionary approach. *Foundations of Computational, IntelligenceVolume 6*, pages 193–207, 2009. doi: https://doi.org/10.1007/978-3-642-01091-0_9.

[9] M. Cuturi and M. Blondel. Soft-DTW: a differentiable loss function for time-series. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 894–903. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/cuturi17a.html`.

[10] B. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster Analysis*. Wiley Series in Probability and Statistics. Wiley, 2011. ISBN 9780470978443.

[11] T.-c. Fu, F.-l. Chung, V. Ng, and R. Luk. Pattern discovery from stock time series using self-organizing maps. In *Workshop Notes of KDD2001 Workshop on Temporal Data Mining*, volume 1. Citeseer, 2001.

[12] X. Golay, S. Kollias, G. Stoll, D. Meier, A. Valavanis, and P. Boesiger. A new correlation-based fuzzy logic clustering algorithm for fmri. *Magnetic resonance in medicine*, 40(2):249–260, 1998.

[13] Google. Welcome to colaboratory. *Google Colaboratory*, 2022. https://colab.research.google.com/.

[14] C. Goutte, P. Toft, E. Rostrup, F. Å. Nielsen, and L. K. Hansen. On clustering fmri time series. *NeuroImage*, 9(3):298–310, 1999.

[15] J. Hämäläinen, S. Jauhiainen, and T. Kärkkäinen. Comparison of internal clustering validation indices for prototype-based clustering. *Algorithms*, 10(3):105, 2017.

[16] K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of arima time-series. In *Proceedings 2001 IEEE international conference on data mining*, pages 273–280. IEEE, 2001.

[17] M. Kendall and J. D. Gibbons. *Rank Correlation Methods*. A Charles Griffin Title, 5 edition, September 1990.

[18] E. Keogh and J. Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and information systems*, 8(2): 154–177, 2005.

[19] E. J. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7: 349–371, 2004.

[20] K. Košmelj and V. Batagelj. Cross-sectional approach for clustering time varying data. *Journal of Classification*, 7(1):99–109, 1990.

[21] M. Kumar, N. Patel, and J. Woo. Clustering seasonality patterns in the presence of errors. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 01 2002. doi: 10.1145/775107.775129.

[22] T. Liao. A clustering procedure for exploratory mining of vector time series. *Pattern Recognition*, 40:2550–2562, 09 2007. doi: 10.1016/j.patcog.2007.01.005.

[23] T. Liao, B. Bolt, J. Forester, E. Hailman, C. Hansen, R. Kaste, and J. O'May. Understanding and projecting the battle state. *23rd Army Science Conference*, December, 2002.

[24] E. Maharaj. Clusters of time series. *Journal of Classification*, 17, 01 2000.

[25] D. Mining. Concepts and techniques. *Jiawei Han and Micheline Kamber*, 2, 2001.

[26] T. Mitsa. *Temporal data mining*. Chapman & Hall/CRC Press Taylor and Francis Group, Chapman & Hall/CRC Press Taylor and Francis Group, 2010.

[27] C. Möller-Levet, F. Klawonn, K.-H. Cho, and O. Wolkenhauer. Fuzzy clustering of short time-series and unevenly distributed sampling points. In *IDA*, volume 2810, pages 330–340, 08 2003. ISBN 978-3-540-40813-0. doi: 10.1007/978-3-540-45231-7_31.

[28] A. Nielsen. *Practical Time Series Analysis: Prediction with Statistics and Machine Learning*. O'Reilly Media, Incorporated, 2019. ISBN 9781492041658. URL `https://books.google.com.mx/books?id=uq0avgEACAAJ`.

[29] V. Niennattrakul and C. Ratanamahatana. Clustering multimedia data using time series. *2006 International Conference on Hybrid Information Technology*, 1:372–379, 2006.

[30] T. Oates, L. Firoiu, and P. R. Cohen. Clustering time series with hidden markov models and dynamic time warping. In *Proceedings of the IJCAI-99 workshop on neural, symbolic and reinforcement learning methods for sequence learning*, pages 17–21. Citeseer, 1999.

[31] L. M. Owsley, L. E. Atlas, and G. D. Bernard. Self-organizing feature maps and hidden markov models for machine-tool monitoring. *IEEE transactions on signal processing*, 45(11):2787–2798, 1997.

[32] T. pandas development team. *pandas-dev/pandas: Pandas 1.0.0*. Zenodo, Jan. 2020. doi: 10.5281/zenodo.3630805. URL `https://doi.org/10.5281/zenodo.3630805`.

[33] E. Patel and D. Kushwaha. Clustering cloud workloads: K-means vs gaussian mixture model. *Procedia Computer Science*, 171:158–167, 01 2020. doi: 10.1016/j.procs.2020.04.017.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[35] D. Piccolo. A distance measure for classifying arima models. *Journal of Time Series Analysis*, 11:153 – 164, 06 2008. doi: 10.1111/j.1467-9892.1990.tb00048.x.

[36] M. Ramoni, P. Sebastiani, and P. Cohen. Bayesian clustering by dynamics. *Machine learning*, 47(1):91–121, 2002.

[37] S. Rani and G. Sikka. Recent techniques of clustering of time series data: a survey. *International Journal of Computer Applications*, 52(15), 2012.

[38] H. Ritchie, E. Mathieu, L. Rodés-Guirao, C. Appel, C. Giattino, E. Ortiz-Ospina, J. Hasell, B. Macdonald, D. Beltekian, and M. Roser. Coronavirus pandemic (covid-19). *Our World in Data*, 2020. https://ourworldindata.org/coronavirus.

[39] D. Roverso. Multivariate temporal classification by windowed wavelet decomposition and recurrent neural networks. *International Topical Meeting on Nuclear Plant Instrumentation, Controls, and Human-Machine Interface Technologies (NPIC&HMIT 2000)*, 11 2000.

[40] J. W. Shavlik, T. Dietterich, and T. G. Dietterich. *Readings in machine learning*. Morgan Kaufmann, 1990.

[41] C. Shaw and G. King. Using cluster analysis to classify time series. *Physica D: Nonlinear Phenomena*, 58(1-4):288–298, 1992.

[42] R. H. Shumway. Time-frequency clustering and discriminant analysis. *Statistics & probability letters*, 63(3):307–314, 2003.

[43] C. Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15:88–103, 1904.

[44] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods. Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020. URL `http://jmlr.org/papers/v21/20-091.html`.

[45] G. Uriel. Timex-clustering. `https://raw.githubusercontent.com/uGR17/TIMEX_CLUSTERING/main/examples/datasets/k_means_example_5ts.csv`, 2022.

[46] G. Uriel. Timex-clustering. `https://raw.githubusercontent.com/uGR17/TIMEX_CLUSTERING/main/examples/datasets/k_means_example_5ts.csv`, 2022.

[47] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

[48] J. Van Wijk and E. Van Selow. Cluster and calendar based visualization of time series data. In *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis'99)*, pages 4–9, 1999. doi: 10.1109/INFVIS.1999.801851.

[49] S. Vishwakarma and V. Lyubchich. Time series clustering and classification. *Technometrics*, 63(3):441–441, 2021. URL `https://doi.org/10.1080/00401706.2021.1945330`.

[50] M. Vlachos, J. Lin, E. Keogh, and D. Gunopulos. A wavelet-based anytime algorithm for k-means clustering of time series. In *In proc. workshop on clustering high dimensionality data and its applications*. Citeseer, 2003.

[51] X. Wang, K. A. Smith, R. Hyndman, and D. Alahakoon. A scalable method for time series clustering. In *Proc. Unrefereed Res. Papers*, volume 1, 2004.

[52] T. Warren Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38 (11):1857–1874, 2005. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2005.01.025.

[53] J. Wilpon and L. Rabiner. A modified k-means clustering algorithm for use in isolated work recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(3):587–594, 1985.

[54] Y. Xiong and D.-Y. Yeung. Mixtures of arma models for model-based time series clustering. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 717–720. IEEE, 2002.

A | Appendix A

TIMEX-CLUSTERING accepts a configuration file specified by the user, used to tune the parameters that are available in TIMEX-CLUSTERING. An example for the Trace dataset, experiment in Section 5.1, is here provided:

```
param_config = {
  "activity_title": "UEA&UCR Dataset - Clustering Example",
  "verbose": "INFO",
  "input_parameters": {
    "source_data_url": "https://raw.githubusercontent.com/uGR17/
   TIMEX_CLUSTERING/main/examples/datasets/k_means_example_5ts.csv",
    "index_column_name": "date",  # Use this column as index
    "frequency": "D",
    "timeseries_names":{
      "date": "Date",
      "ts1": "timeseries1",
      "ts2": "timeseries2",
      "ts3": "timeseries3",
    }
  },
  "model_parameters": {
     "clustering_approach": "observation_based,feature_based,model_based
   ",
     "models": "k_means,gaussian_mixture",
     "distance_metric": "euclidean,dtw,softdtw",
     "feature_transformations": "DWT",
     "n_clusters": [3, 4, 5, 6],
     "gamma": 0.01,
     "main_accuracy_estimator": "silhouette"
  },
  "visualization_parameters": {
    "xcorr_graph_threshold": 0.8,
  }
}
```

Listing A.1: Parameter configuration dictionary for the Trace dataset.

An example of the configuration file for the COVID-19 dataset, experiment in Section 5.2, is here provided:

```
param_config_covid = {
  "activity_title": "COVID-19 clustering with TIMEX-CLUSTERING framework ",
  "verbose": "INFO",
  "input_parameters": {
    "source_data_url": "https://raw.githubusercontent.com/uGR17/TIMEX_CLUSTERING/main/examples/datasets/World_Covid_Dataset.csv",
    "index_column_name": "date",  # Use this column as index
    "frequency": "D",
  },
  "model_parameters": {
     "clustering_approach": "observation_based,feature_based,model_based ",
     "models": "k_means,gaussian_mixture",
     "pre_transformation": "log_modified",
     "distance_metric": "euclidean,dtw",
     "feature_transformations": "DWT",
     "n_clusters": [3, 4, 5],
     "gamma": 0.01,
     "main_accuracy_estimator": "silhouette"
  },
  "visualization_parameters": {
    "xcorr_graph_threshold": 0.8,
  }
}
```

Listing A.2: PParameter configuration dictionary for the COVID-19 dataset.

Code used to create the interactive website for the Trace dataset.

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import pandas as pd
import plotly.graph_objs as go
from timexseries_clustering.data_visualization.functions import create_timeseries_dash_children
from jupyter_dash import JupyterDash
from dash.dependencies import Input, Output

children_for_each_timeseries = [{
    'name': s.approach,
    'children': create_timeseries_dash_children(s, param_config_kmeans)
```

```
} for s in timeseries_containers]

app = JupyterDash(__name__)

disclaimer = [html.Div([
    html.H1("Observation based Clustering example", style={'text-align': 'center'}),
    html.Hr(),
    html.Div([html.Img(src = 'https://raw.githubusercontent.com/uGR17/TIMEX_CLUSTERING/main/examples/figures/Politecnico-di-Milano.jpg',
                       style = {'padding-top': '1%','height': '32%','width': '32%'})], style={'textAlign': 'center'}),
    html.H4(
        "Dashboard by the Intelligent Embedded Systems (IES) research group of the Politecnico di Milano, Italy",
        style={'text-align': 'center', 'top-margin': '25px'}),
    html.Hr(),
    dcc.Markdown('''
        Welcome to the monitoring and clustering dashboard! Provided by the Intelligent Embedded Systems (IES) research group of Politecnico di Milano, Italy.

        The dashboard relies on TIMEX-CLUSTERING, a Python-based framework for automatic time series analysis developed by the IES research group.

        You can select the data to be visualized from the selector at the bottom of the page.
        '''),
    html.Br(),
    html.H2("Please select the clustering approach:")
    ], style={'width': '80%', 'margin': 'auto'}),
dcc.Dropdown(
    id='timeseries_selector',
    options=[{'label': i['name'], 'value': i['name']} for i in children_for_each_timeseries],
    value='Time-series'
), html.Div(id="timeseries_wrapper"), html.Div(dcc.Graph(), style={'display': 'none'})]
tree = html.Div(children=disclaimer, style={'width': '80%', 'margin': 'auto'})
app.layout = tree

@app.callback(
```

```
45      Output(component_id='timeseries_wrapper', component_property='
        children'),
46      [Input(component_id='timeseries_selector', component_property='value
        ')])
47  def update_timeseries_wrapper(input_value):
48      try:
49          children = next(x['children'] for x in
        children_for_each_timeseries if x['name'] == input_value)
50      except StopIteration:
51          return html.Div(style={'padding': 200})
52      return children
53
54  app.run_server(mode='external')#inline-external
```

Listing A.3: Code to create the interactive website for the clustering of Trace dataset.

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank my supervisors, Ph.D Manuel Roveri and Ph.D student Alessandro Falcetta, for allowing me the honor to do the thesis with them and for sharing their enthusiasm, ideas and passion throughout the development of this thesis. Every time, they inspired me to push my work further, I also appreciate the resources and examples provided, as Timex, that prompted me to create a very good quality new framework. I will miss our Team meetings.

The biggest thanks goes to my parents and my family, who supported me in every plan I had, since the first moment that I decided to travel from Mexico to Italy to study my master's degree, you have always been throughout my life, supporting me and encouraging me to achieve and fulfill my dreams. Without them I would not have achieved half of what I have done in my life. I love you.

Finally, I want to thank every person I have the pleasure to meet in these years in Italy. I wish I could name them all, friends, professors, strangers, but all you, from Pennsylvania to Bengaluru, from Paris to Sicily, from China to Puglia, thank you, I learned a lot these past years and thanks to have made this journey through university much more than just studying.